Continue

Enjoy sharper detail, more accurate color, lifelike lighting, believable backgrounds, and more with our new model update. Your generated images will be more polished thanever.See What's NewExplore how consumers want to see climate stories told today, and what that means for yourvisuals.Download Our Latest VisualGPS ReportData-backed trends. Generative AI demos. Answers to your usage rights questions. Our original video podcast covers it allnow ondemand.Watch NowEnjoy sharper detail, more accurate color, lifelike lighting, believable backgrounds, and more with our new model update. Your generated images will be more polished thanever.See What's NewExplore how consumers want to see climate stories told today, and what that means for yourvisuals.Download Our Latest VisualGPS ReportData-backed trends. Generative AI demos. Answers to your usage rights questions. Our original video podcast covers it allnow ondemand.Watch NowEnjoy sharper detail, more accurate color, lifelike lighting, believable backgrounds, and more with our new model update. Your generated images will be more polished thanever.See What's NewExplore how consumers want to see climate stories told today, and what that means for yourvisuals.Download Our Latest VisualGPS ReportData-backed trends. Generative AI demos. Answers to your usage rights questions. Our original video podcast covers it allnow ondemand.Watch NowWelcome to the comprehensive guide on IB Computer Science. This course offers high school students an opportunity to delve into coding, algorithms, and data science, equipping them with essential skills for the future. In this guide, we'll demystify the IB Computer Science syllabus, explain the grading system, provide preparation tips, and explore potential career paths. Whether you're considering this subject or ready to embark on this journey, let's explore the dynamic world of IB Computer Science together.The International Baccalaureate (IB) Computer Science examinations for May 2025 are scheduled as follows:Friday, 2 May 2025 - Afternoon:Computer Science HL Paper 1: 2 hours 10 minutesComputer Science SL Paper 1: 1 hour 30 minutesMonday, 5 May 2025 - Morning:Computer Science HL Paper 2: 1 hour 20 minutesComputer Science HL Paper 3: 1 hourComputer Science SL Paper 2: 1 hourThese exams assess students' understanding of computational thinking, problem-solving, and programming concepts. It's crucial to familiarise yourself with the specific content and structure of each paper to ensure thorough preparation. Check out IB Exam Schedule 2025 for more subjects!In today's technology-driven world, Computer Science is a highly sought-after discipline. The IB Computer Science course provides an in-depth study of both theoretical and practical aspects, laying the foundation for a promising tech career.Benefits of Studying IB Computer Science:Early Advantage: Studying computer science at the school level enhances problem-solving skills and logical reasoning, preparing students for the digital age.Real-World Application: The course emphasizes real-world problem-solving, algorithmic thinking, and software development, equipping students with essential skills.High Demand: In the UK alone, there's an annual demand for 124,000 engineers and technicians with core engineering skills, highlighting the importance of this field.Transferable Skills: The versatility of computer science ensures that the skills gained are applicable across various sectors.In the words of Bill Gates, Its an amazing time to be a student in computing, as the boundaries of what computers can do are continually being pushed. Whether you're passionate about building technology, analyzing data, or solving global challenges, IB Computer Science offers an intellectually stimulating and career-enhancing pathway.IB Computer Science is often considered one of the Hardest IB Subjects. It is challenging but rewarding, requiring strong problem-solving skills, logical thinking, and a solid grasp of programming concepts. While some students find it manageable, others may struggle, especially if they lack prior coding experience.Comparing IB Computer Science SL vs HL DifficultyStandard Level (SL)Higher Level (HL)WorkloadModerateSignificantly higherCoding ComplexityCovers basic programming conceptsIncludes advanced algorithms and data structuresMathematical DemandLowerHigher (more focus on computational thinking)Assessment Components2 exam papers + Internal Assessment3 exam papers + Internal AssessmentRequired Programming KnowledgeFundamental coding skillsStronger programming proficiency requiredOverall DifficultyManageable with consistent practiceMore demanding, requires deeper understandingWhat Makes IB Computer Science Challenging? Algorithmic Thinking: Students must develop strong logical and computational thinking skills to solve complex problems.Programming Mastery: Proficiency in at least one programming language (Java, Python, or JavaScript) is essential.Theory & Application: The course requires a balance of theoretical concepts (networking, databases, cybersecurity) and practical coding skills.Time-Intensive Internal Assessment (IA): Students must complete an independent coding project, which requires careful planning and implementation.Is IB Computer Science Right for You? Choose SL if: Youre interested in technology but prefer a less intensive workload or have limited programming experience. Choose HL if: You enjoy coding, problem-solving, and mathematical logic, and possibly plan to pursue computer science, engineering, or data science in university.Hannah Craig, IB Computer Science HL student who is now a Software Engineer at a leading tech company had the following to say:"Studying IB Computer Science was indeed a challenging but rewarding journey. What really ensured my success was a blend of consistent effort, curiosity, and problem-solving mindset. I found that regular practice, particularly with coding, was crucial. Don't just limit yourself to the classroom. Explore online coding platforms, participate in competitions, and most importantly, try to solve real-world problems with the skills you're learning. This not only deepens your understanding but also keeps the process exciting. Always remember, Computer Science is not just about coding, it's about leveraging technology to make life better."The IB Computer Science syllabus has been updated for first teaching in 2025 and first assessment in 2027, with a greater focus on computational thinking, problem-solving, and modern technologies like machine learning. The course is now structured into two major themes, a case study, and an internal assessment.IB Computer Science Syllabus Structure (2025)ComponentDescriptionSL HoursHL HoursTheme A: Concepts of Computer ScienceCovers foundational computer science topics, including networking, databases, and machine learning.105195Theme B: Computational Thinking & Problem-SolvingFocuses on programming, object-oriented programming (OOP), and algorithmic problem-solving.3872Case StudyAnalyzes an emerging technology topic in depth.1530Internal AssessmentStudents develop a computational solution to a real-world problem.3535Collaborative Sciences ProjectA cross-disciplinary project encouraging teamwork and application of computing concepts.1010Total Teaching Hours150240Theme A: Concepts of Computer ScienceThis section develops fundamental knowledge required in computing:A.1 Computer Fundamentals: Includes hardware, software, system architecture, and ethical issues.A.2 Networks: Covers network structures, protocols, cybersecurity, and cloud computing.A.3 Databases: Explores SQL, data storage, and relational database models.A.4 Machine Learning (New for 2025!): Introduces AI, algorithms, and data-driven decision-making.Theme B: Computational Thinking & Problem-SolvingThis section focuses on coding and problem-solving, refining students' logical and algorithmic skills.B.1 Computational Thinking: Understanding problem decomposition, abstraction, and algorithm efficiency.B.2 Programming: Covers variables, control structures, functions, and debugging techniques.B.3 Object-Oriented Programming (OOP): Introduces classes, objects, inheritance, and encapsulation.B.4 Abstract Data Types (HL Only): Covers stacks, queues, linked lists, and trees for advanced problem-solving.Case Study (New for 2025!)The IB has introduced a Case Study to allow students to explore an emerging technology in-depth, such as:The impact of quantum computing on encryptionEthical concerns in AI and automationAdvances in biocomputing and DNA storageThis component develops research and analytical skills, preparing students for university-level study.The Internal Assessment (IA) Weighting: 20% of the final IB Computer Science grade for both SL and HL. Purpose: Students design, develop, and evaluate a computational solution to a real-world problem. Key Requirements:Research & Planning: Identify a problem and propose a computing-based solution.Implementation: Develop a functional program using a programming language (e.g., Python, Java).Evaluation: Reflect on strengths, weaknesses, and areas for improvement.The IA is an excellent opportunity to apply computational thinking skills in a real-world context, fostering innovation and technical proficiency. For more tips on how to tackle the IA, read on to find out. As Professor Tim Bell, a leading computer science educator, highlights, the updated IB Computer Science syllabus provides a well-rounded approach, combining core computing principles, hands-on programming, and real-world applications. Whether you're interested in AI, data science, or software development, this syllabus equips students with the skills needed for further study and careers in tech. Need help mastering IB Computer Science? Our expert IB Computer Science tutors can guide you through the syllabus, IA, and exam preparation! Choosing between IB Computer Science and A-Level Computer Science depends on your learning style, academic goals, and career aspirations. While both qualifications cover essential computing concepts, they differ significantly in structure, content focus, and assessment methods.Key Differences Between IB and A-Level Computer ScienceIB Computer ScienceA-Level Computer ScienceCourse StructureTwo themes: Concepts of Computer Science and Computational Thinking & Problem-Solving, plus a Case Study and Internal Assessment.Two components: Theory and Non-Examined Assessment (Programming Project).Programming FocusProgramming is part of Theme B, with additional coverage in Object-Oriented Programming (OOP) (HL).40% of the course is programming, with an emphasis on algorithms, problem-solving, and computational thinking.Content BreadthCovers a broader range of topics, including networks, databases, machine learning, and cybersecurity.More focused on computer architecture, algorithms, and programming principles.Assessment StyleCombination of written exams, internal assessments (IA), and coursework.Primarily exam-based, with a separate programming project contributing to the final grade.Final ProjectInternal Assessment: Students develop a computational solution for a real-world problem.Non-Examined Assessment (NEA): A programming project focused on problem-solving.Global RecognitionRecognized by international universities, especially for technology-related degrees.Primarily recognized in UK universities, but also accepted internationally.Which One is Harder? Exam Performance Statistics Exam results indicate that achieving the highest grade is more common in IB Computer Science than in A-Level Computer Science:IB Computer Science HL: 8.2% of students achieved a 7/7.A-Level Computer Science: 9.3% of students achieved an A*.This suggests that A-Level Computer Science may offer a slightly better chance of securing top marks, particularly due to the coursework component (IA) in IB which can increase students' workload. Which One Should You Choose? Choose IB Computer Science if:You prefer a broad curriculum that covers theory, programming, and emerging technologies like machine learning.You want a mix of coursework (IA) and exams, rather than purely exam-based assessment.You plan to apply to international universities, as IB is widely accepted worldwide. Choose A-Level Computer Science if:You prefer a deeper focus on programming and computational problem-solving.You are comfortable with mainly exam-based assessments, with one large programming project.You plan to apply to UK universities, where A-Levels are the standard qualification. Need expert guidance? Our IB Computer Science tutors & A-Level Computer Science tutors can help you excel in either course and maximize your university admissions success! Have a look at our comprehensive set of IB Computer Science SL and IB Computer Science HL resources, developed by expert IB teachers and examiners: View IB Computer Science SL Study Notes View IB Computer Science HL Study Notes View IB Computer Science SL Past Papers View IB Computer Science HL Past PapersWhen deciding between Standard Level (SL) and Higher Level (HL) in IB Computer Science, it's essential to consider your interests, academic goals, and future aspirations. Here's a detailed comparison based on the IBO guide to assist your decision-making:Course Structure and Content:Standard Level (SL):Provides a foundational understanding of computer science principles.Covers core topics such as system fundamentals, computer organization, networks, and basic programming concepts.Total teaching hours: 150.Higher Level (HL):Offers an in-depth exploration of computer science.Includes all SL topics plus additional content like abstract data structures, resource management, and control.Emphasizes computational thinking and advanced problem-solving skills.Total teaching hours: 240.Assessment Components:External Assessments:Both SL and HL students undertake written examinations covering the core syllabus.HL students face additional papers that assess their understanding of the extended topics.Weighs more for HL (80%), compared to SL (70%) Internal Assessment:Both levels require a project where students develop a computational solution to a real-world problem.This project is more demanding at the HL, reflecting the advanced content covered.Weighs more for SL (30%), compared to HL (20%) Considerations for Decision-Making:Interest and Commitment: If you have a strong passion for computer science and are willing to engage deeply with complex topics, HL might be suitable. For those looking to gain a solid understanding without the additional workload, SL could be appropriate.Future Academic and Career Plans: Students aiming for university programs in computer science or related fields may benefit from the comprehensive preparation that HL offers. If computer science is not your primary focus for higher education, SL provides a balanced approach.Workload and Time Management: HL requires a greater time investment due to its expanded curriculum and assessment components. Consider your overall IB subject load and extracurricular commitments when deciding.Your choice between SL and HL should align with your personal interests, academic goals, and capacity to manage the coursework. Both levels offer valuable skills and knowledge, but HL provides a more rigorous and in-depth study suitable for those considering a future in computer science.Number of IB Computer Science SL & HL candidates in 2024Understanding how IB Computer Science is graded helps you strategize your efforts and maximize your score. Like all IB subjects, it is scored out of 7 points, with additional points available from the Theory of Knowledge (TOK) and Extended Essay (EE) components.How Is IB Computer Science Graded?ComponentWeightingDescriptionExternal Assessment80%Written exams testing core topics, computational thinking, and case study analysis.Internal Assessment (IA)20%A programming project where students develop a computational solution to a real-world problem.Breakdown of IB Computer Science External AssessmentPaperSL DurationHL DurationContent CoveredWeighting (SL)Weighting (HL)Paper 11h 30m2h 10mCore topics, computational thinking, and problem-solving.45%40%Paper 21h 1h 20mCase study analysis and application of concepts.25%20%Paper 3 (HL only)-1hAdvanced computational thinking & emerging tech case study.N/A20%Paper 1: Focuses on core computer science topics, algorithms, and computational thinking.Paper 2: Requires students to analyze a case study and apply problem-solving skills.Paper 3 (HL only): Assesses advanced concepts and the application of emerging technologies.Internal Assessment (IA) The Programming Project Weighting: 20% of the final IB Computer Science grade. Project Requirements: Students design, develop, and evaluate a computational solution to a real-world problem. Assessment Criteria:Planning & Design Identifying a problem and proposing a structured solution.Development & Implementation Writing and refining the program.Evaluation Assessing strengths, limitations, and improvements.What Are the Grade Boundaries for IB Computer Science? Grade boundaries vary each year, but historical data shows:GradeApproximate Score Required (%)785%+675-84%565-74%450-64%340-49%230-39%1Below 30%To achieve a high grade in IB Computer Science, students must balance strong theoretical understanding (External Assessment) with hands-on programming skills (Internal Assessment). Regular practice with past papers, solid programming fundamentals, and a well-executed IA project can significantly improve your chances of scoring a 7. Need expert guidance? Our IB Computer Science tutors can help refine your exam technique, programming skills, and IA project for maximum success!The updated IB Computer Science exam structure (first assessment in 2027) is designed to assess both theoretical knowledge and practical problem-solving skills. It consists of written papers and an Internal Assessment (IA), with additional requirements for HL students.IB Computer Science Exam Structure (2025)SL DurationHL DurationContent CoveredWeighting (SL)Weighting (HL)Paper 11h 30m2h 10mCore topics: Computer Fundamentals, Networks, Databases, and Machine Learning.45%40%Paper 21h1h 20mComputational Thinking & Programming, including problem-solving tasks.25%20%Paper 3 N/A1hCase Study: Analyzing and applying emerging technologies to real-world problems.N/A20%Detailed Breakdown of Exam ComponentsPaper 1 (SL & HL):Assesses core concepts in computer science, including computer fundamentals, networks, databases, and machine learning.Includes multiple-choice, short-answer, and extended-response questions.Paper 2 (SL & HL):Focuses on computational thinking and programming, testing students' ability to analyze and solve coding problems.Entirely consists of extended-response questions requiring problem-solving strategies.Paper 3 (HL Only New Case Study Component):Introduces a real-world emerging technology case study, such as AI, blockchain, or quantum computing.Requires students to analyze technological implications, propose solutions, and evaluate computing advancements.Internal Assessment (IA) Programming Project Weighting: 20% of the final grade for both SL and HL. Objective: Develop a computational solution to a real-world problem using a programming language (Python, Java, etc.). Assessment Criteria:Problem Identification & Research Define a clear computing problem with real-world applications.Solution Development Write, test, and refine a working program.Evaluation & Documentation Analyze the effectiveness, limitations, and improvements of the project.Need help preparing for IB Computer Science? Our IB Computer Science Tutors can provide guidance on exam techniques, programming skills, and IA project development to help you secure top marks! For the timeline of this year's IB Computer Science exam, check out the IB Exam Schedule 2025. A good score in IB Computer Science depends on your goals and university requirements. While a 4 is a passing grade, 6 or 7 is considered high achievement and ideal for competitive programs. Recent Performance Data (2024):According to the IB Diploma Programme Statistical Bulletin for May 2024, the grade distribution for Computer Science Higher Level (HL) was as follows:IB Computer Science SL & HL grade distributions in 2024In the May 2024 session, only 8.2% of HL candidates achieved a 7, while an ever lower portion (6.2%) of SL candidates achieved the same score. Meanwhile, the mean grade for Computer Science HL was approximately 4.5. Showing that a grade above that level is satisfactory. However, students should also check their university course requirements. Number and mean grades of IB Computer Science SL & HL candidates in 2024University Admission Requirements:For students aiming to pursue Computer Science at competitive universities, higher IB scores are often expected. For instance:University of Cambridge: Offers typically require an IB score of 41-42 points, with 776 at Higher Level. Durham University: Requires an IB Diploma score of 38, with 766 in Higher Level subjects, including Mathematics.If you're wondering about Cambridge University Admission Requirements, check out our separate article. Aligning Scores with Goals:Personal Academic Goals: Achieving a score of 6 or 7 demonstrates a strong grasp of Computer Science concepts and can be advantageous for university applications and scholarships.University Admissions: Research the specific IB score requirements of your preferred institutions to ensure your scores align with their expectations.Career Aspirations: High IB scores can reflect proficiency and dedication, qualities valued in both academic and professional settings.In summary, while a score of 4 is passing, aiming for a 6 or 7 is advisable, especially if you plan to study Computer Science at a competitive university. Ultimately, a "good" IB Computer Science score is one that aligns with and supports your individual academic and career objectives.Scoring a 7 in IB Computer Science, the highest grade, requires consistent effort, smart study strategies, and an in-depth understanding of the subject matter. Here are some proven strategies to help you reach that goal:Understand the Syllabus: The IBO syllabus is your roadmap. Knowing it inside out ensures you are aware of what's required and can direct your studies effectively.Understand Concepts, Don't Just Memorise: Computer science is a practical subject, so understanding concepts and knowing how to apply them is crucial. One useful resource is IB Computer Science HL Study Notes and IB Computer Science SL Study Notes.Get help from anIB Computer Science tutor: If you're struggling with a particular topic, it's a good idea to get help from a tutor. A tutor can provide one-on-one support and guidance, and help you to overcome any areas of difficulty.Hands-On Practice: Use online platforms like Codecademy or HackerRank to practice coding skills regularly. This aids in understanding the practical aspects of the course and helps in the Internal Assessment.Internal Assessment: Dedicate sufficient time to your IA project. It contributes 20% of your final grade. Dr. Sue Sentance suggests choosing a problem you're genuinely interested in solving; it makes the process more engaging and the outcome more successful.Utilise Available Resources: There are numerous online resources to help in your revision. Sites like IBO and IB CompSciHub offer a wealth of study materials.Remember, it's a marathon, not a sprint. Consistent effort over the two-year course is more likely to earn you a 7 than last-minute cramming.The IB Computer Science Internal Assessment (IA) is a critical component of the course, contributing 20% of the final grade for both SL and HL students. The IA takes the form of a computational solution to a real-world problem and is assessed by the following four key stages Planning, Design, Development, and Evaluation. Python, Java, C++, or any language suited to the project is welcome. IB Computer Science IA Structure (2025)StageDescriptionKey Deliverables1 Planning & InvestigationIdentify a real-world problem, define the target audience and client needs, and justify why a computational solution is necessary.Problem Statement, Feasibility Study, Success Criteria2 DesignCreate detailed system diagrams, user interface sketches, flowcharts, and test plans. Outline data structures, algorithms, and validation techniques.Flowcharts, UML Diagrams, Pseudocode, Test Plan3 DevelopmentImplement the software solution using a programming language. Ensure proper coding structure, debugging, and documentation.Fully Functional Code, Testing Logs, Annotated Code4 EvaluationAssess the effectiveness of the solution based on the success criteria. Gather client/user feedback and propose improvements.User Feedback, Self-Reflection, Potential EnhancementsTips for Excelling in the IB Computer Science IAChoose a Meaningful Problem: Select a real-world problem that is engaging and practical to solve.Plan Your Time Wisely: Start early to allow time for debugging and refining your project.Showcase Your Coding Skills: Use efficient algorithms and structured programming practices.Provide Detailed Documentation: Clearly explain how your solution meets user needs and why design choices were made.Test Rigorously: Include edge cases, error handling, and validation techniques.The IB Computer Science IA is an excellent opportunity to apply computational thinking and software development skills in a practical project. By following the four IA stages, structuring your documentation effectively, and demonstrating problem-solving abilities, you can maximize your score and strengthen your university applications.To excel in IB Computer Science, its essential to use high-quality study materials and engage with interactive learning platforms. Below is a curated list of the top resources recommended by professional IB tutors to help you prepare effectively. Official IB Resources IBOs Official Computer Science GuideCovers the latest syllabus, assessment objectives, and ethical guidelines.Essential for understanding exam expectations and structuring your study plan. IB Examiner Reports & Mark SchemesProvides insights into how examiners grade responses.Helps identify common mistakes and high-scoring techniques.Best IB Computer Science Textbooks "Computer Science for the IB Diploma" (Cambridge University Press)Comprehensive coverage of all syllabus topics.Includes exam-style questions and worked examples. "IB Computer Science Study & Revision Guide"Focused on concept explanation and exam techniques.Ideal for quick revision and topic summaries.Online Learning Platforms for IB Computer Science Codecademy Interactive tutorials on Python, Java, and Data Structures. Khan Academy Clear explanations of algorithms, networking, and computational thinking. HackerRank Coding challenges to improve problem-solving skills.IB-Specific Study Hubs & IA Support IB CompSciHubProvides topic summaries, IA examples, and past paper solutions. IB Revision VillageExtensive collection of IB past papers and practice questions.IB Past Paper Practice The Key to Exam Success Why it matters: Practicing past papers is the most effective way to prepare for IB exams. Recommended Platforms:IBO Past Papers Authentic past exam papers with official mark schemes.Personalised Tutoring & IB Computer Science Q&A Notes TutorChase IB Computer Science Tutors1-on-1 tutoring for exam preparation, IA guidance, and concept clarity.Includes personalized study plans and expert feedback. IB Computer Science Q&A Revision NotesConcise, exam-focused notes covering key concepts and problem-solving techniques.According to a 2024 report by the UK's Department for Digital, Culture, Media and Sport (DCMS), the digital sector grew four to five times faster than any other sector in the UK from 2019 to 2022. With the sector's rapid growth, the demand for computer science professionals is on a significant rise which means the skills and knowledge gained from studying IB Computer Science can open doors to a plethora of exciting and lucrative career opportunities: Software Development: Armed with an IB Computer Science background, you can become a software developer or a web developer, creating programs, websites, or mobile apps. Companies from startups to tech giants like Google and Amazon are always on the hunt for talented developers.Data Analysis: With the rise of big data, companies need data analysts to make sense of the massive amounts of information they collect. The logical and mathematical skills you gain from computer science can help you excel in this field.Cybersecurity: With increasing digital threats, cybersecurity has become a significant concern for businesses and governments worldwide. Professionals in this field work to protect systems and data from cyber-attacks.AI and Machine Learning: AI is at the forefront of technology today. Roles in this field involve creating and using algorithms to help machines learn from data.Studying IB Computer Science doesn't just prepare you for tech-related roles. According to Tech Nation, a significant portion of the digital tech jobs are in non-tech industries, such as finance and health.IB Computer Science builds computational thinking and problem-solving skills, preparing students for careers in programming, cybersecurity, and data science. This guide provides key insights, revision strategies, and top resources to help you succeed. Beyond grades, the real value lies in the skills and knowledge you develop for the digital future.Study of computationFor other uses, see Computer science (disambiguation).Fundamental areas of computer scienceProgramming language theoryComputer architectureArtificial intelligenceComputational complexity theoryComputer scienceHistoryOutlineGlossaryCategoryvteComputer science is the study of computation, information, and automation.[1][2][3] Computer science spans theoretical disciplines (such as algorithms, theory of computation, and information theory) to applied disciplines (including the design and implementation of hardware and software).[4][5][6] Algorithms and data structures are central to computer science.[7]The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and preventing security vulnerabilities. Computer graphics and computational geometry address the generation of images. Programming language theory considers different ways to describe computational processes, and database theory concerns the management of repositories of data. Humancomputer interaction investigates the interfaces through which humans and computers interact, and software engineering focuses on the design and principles behind developing software. Areas such as operating systems, networks and embedded systems investigate the principles and design behind complex systems. Computer architecture describes the construction of computer components and computer-operated equipment. Artificial intelligence and machine learning aim to synthesize goal-orientated processes such as problem-solving, decision-making, environmental adaptation, planning and learning found in humans and animals. Within artificial intelligence, computer vision aims to understand and process image and video data, while natural language processing aims to understand and process textual and linguistic data.The fundamental concern of computer science is determining what can and cannot be automated.[2][8][3][9][10] The Turing Award is generally recognized as the highest distinction in computer science.[11][12]Main article: History of computer scienceHistory of computingHardwareHardware 1960s to presentSoftwareSoftware configuration managementUnixFree software and open-source softwareComputer scienceArtificial intelligenceCompiler constructionEarly computer scienceOperating systemsProgramming languagesProminent pioneersSoftware engineeringModern conceptsGeneral-purpose CPUsGraphical user interfaceInternetLaptopsPersonal computersVideo gamesWorld Wide WebCloudQuantumBy countryBulgariaEastern BlocPolandRomaniaSouth AmericaSoviet UnionYugoslaviaTimeline of computingbefore 1950195019791980198919901999200020092010201920 20presentmore timelines ...Glossary of computer science CategoryvteGottfried Wilhelm Leibniz (16461716) developed logic in a binary number system and has been called the "founder of computer science".[13]Charles Babbage is sometimes referred to as the "father of computing".[14] Ada Lovelace published the first algorithm intended for processing on a computer.[15] The earliest foundations of what would become computer science predate the invention of the modern digital computer. Machines for calculating fixed numerical tasks such as the abacus have existed since antiquity, aiding in computations such as multiplication and division. Algorithms for performing computations have existed since antiquity, even before the development of sophisticated computing equipment.[16]Wilhelm Schickard designed and constructed the first working mechanical calculator in 1623.[17] In 1673, Gottfried Leibniz demonstrated a digital mechanical calculator, called the Stepped Reckoner.[18] Leibniz may be considered the first computer scientist and information theorist, because of various reasons, including the fact that he documented the binary number system. In 1820, Thomas de Colmar launched the mechanical calculator industry[note 1] when he invented his simplified arithmometer, the first calculating machine strong enough and reliable enough to be used daily in an office environment. Charles Babbage started the design of the first automatic mechanical calculator, his Difference Engine, in 1822, which eventually gave him the idea of the first programmable mechanical calculator, his Analytical Engine.[19] He started developing this machine in 1834, and "in less than two years, he had sketched out many of the salient features of the modern computer".[20] "A crucial step was the adoption of a punched card system derived from the Jacquard loom"[20] making it infinitely programmable.[note 2] In 1843, during the translation of a French article on the Analytical Engine, Ada Lovelace wrote, in one of the many notes she included, an algorithm to compute the Bernoulli numbers, which is considered to be the first published algorithm ever specifically tailored for implementation on a computer.[21] Around 1885, Herman Hollerith invented the tabulator, which used punched cards to process statistical information; eventually his company became part of IBM. Following Babbage, although unaware of his earlier work, Percy Ludgate in 1909 published[22] the 2nd of the only two designs for mechanical analytical engines in history. In 1914, the Spanish engineer Leonardo Torres Quevedo published his Essays on Automatics,[23] and designed, inspired by Babbage, a theoretical electromechanical calculating machine which was to be controlled by a read-only program. The paper also introduced the idea of floating-point arithmetic.[24][25] In 1920, to celebrate the 100th anniversary of the invention of the arithmometer, Torres presented in Paris the Electromechanical Arithmometer, a prototype that demonstrated the feasibility of an electromechanical analytical engine,[26] on which commands could be typed and the results printed automatically.[27] In 1937, one hundred years after Babbage's impossible dream, Howard Aiken convinced IBM, which was making all kinds of punched card equipment and was also in the calculator business[28] to develop his giant programmable calculator, the ASCC/Harvard Mark I, based on Babbage's Analytical Engine, which itself used cards and a central computing unit. When the machine was finished, some hailed it as "Babbage's dream come true".[29]During the 1940s, with the development of new and more powerful computing machines such as the AtanasoffBerry computer and ENIAC, the term computer came to refer to the machines rather than their human predecessors.[30] As it became clear that computers could be used for more than just mathematical calculations, the field of computer science broadened to study computation in general. In 1945, IBM founded the Watson Scientific Computing Laboratory at Columbia University in New York City. The renovated fraternity house on Manhattan's West Side was IBM's first laboratory devoted to pure science. The lab is the forerunner of IBM's Research Division, which today operates research facilities around the world.[31] Ultimately, the close relationship between IBM and Columbia University was instrumental in the emergence of a new scientific discipline, with Columbia offering one of the first academic-credit courses in computer science in 1946.[32] Computer science began to be established as a distinct academic discipline in the 1950s and early 1960s.[33][34] The world's first computer science degree program, the Cambridge Diploma in Computer Science, began at the University of Cambridge Computer Laboratory in 1953. The first computer science department in the United States was formed at Purdue University in 1962.[35] Since practical computers became available, many applications of computing have become distinct areas of study in their own rights.See also: History of computing and History of informaticsSee also: Informatics EtymologyAlthough first proposed in 1956,[36] the term "computer science" appears in a 1959 article in Communications of the ACM,[37]in which Louis Fein argues for the creation of a Graduate School in Computer Sciences analogous to the creation of Harvard Business School in 1921.[38] Louis justifies the name by arguing that, like management science, the subject is applied and interdisciplinary in nature, while having the characteristics typical of an academic discipline.[37] This effort, and those of others such as numerical analyst George Forsythe, were successful, and universities went on to create such departments, starting with Purdue in 1962.[39] Despite its name, a significant amount of computer science does not involve the study of computers themselves. Because of this, several alternative names have been proposed.[40] Certain departments of major universities prefer the term computing science, to emphasize precisely that difference. Danish scientist Peter Naur suggested the term datalogy,[41] to reflect the fact that the scientific discipline revolves around data and data treatment, while not necessarily involving computers. The first scientific institution to use the term was the Department of Datalogy at the University of Copenhagen, founded in 1969, with Peter Naur being the first professor in datalogy. The term is used mainly in the Scandinavian countries. An alternative term, also proposed by Naur, is data science; this is now used for a multi-disciplinary field of data analysis, including statistics and databases.In the early days of computing, a number of terms for the practitioners of the field of computing were suggested (albeit facetiously) in the Communications of the ACMturingineer, turologist, flow-charts-man, applied meta-mathematician, and applied epistemologist.[42] Three months later in the same journal, comptologist was suggested, followed next year by hypologist.[43] The term computics has also been suggested.[44] In Europe, terms derived from contracted translations of the expression "automatic information" (e.g. "informazione automatica" in Italian) or "information and mathematics" are often used, e.g. informatique (French), Informatik (German), informatica (Italian, Dutch), informtica (Spanish, Portuguese), informatika (Slavic languages and Hungarian) or pliroforiki (, which means informatics) in Greek. Similar words have also been adopted in the UK (as in the School of Informatics, University of Edinburgh).[45] "In the U.S., however, informatics is linked with applied computing, or computing in the context of another domain."[46]A folkloric quotation, often attributed tobut almost certainly not first formulated byEdsger Dijkstra, states that "computer science is no more about computers than astronomy is about telescopes."[note 3] The design and deployment of computers and computer systems is generally considered the province of disciplines other than computer science. For example, the study of computer hardware is usually considered part of computer engineering, while the study of commercial computer systems and their deployment is often called information technology or information systems. However, there has been exchange of ideas between the various computer-related disciplines. Computer science research also often intersects other disciplines, such as cognitive science, linguistics, mathematics, physics, biology, Earth science, statistics, philosophy, and logic.Computer science is considered by some to have a much closer relationship with mathematics than many scientific disciplines, with some observers saying that computing is a mathematical science.[33] Early computer science was strongly influenced by the work of mathematicians such as Kurt Gdel, Alan Turing, John von Neumann, Rzsa Pter and Alonzo Church and there continues to be a useful interchange of ideas between the two fields in areas such as mathematical logic, category theory, domain theory, and algebra.[36]The relationship between computer science and software engineering is a contentious issue, which is further muddied by disputes over what the term "software engineering" means, and how computer science is defined.[47] David Parnas, taking a cue from the relationship between other engineering and science disciplines, has claimed that the principal focus of computer science is studying the properties of computation in general, while the principal focus of software engineering is the design of specific computations to achieve practical goals, making the two separate but complementary disciplines.[48]The academic, political, and funding aspects of computer science tend to depend on whether a department is formed with a mathematical emphasis or with an engineering emphasis. Computer science departments with a mathematics emphasis and with a numerical orientation consider alignment with computational science. Both types of departments tend to make efforts to bridge the field educationally if not across all research.Main article: Philosophy of computer scienceDespite the word science in its name, there is debate over whether or not computer science is a discipline of science,[49] mathematics,[50] or engineering.[51] Allen Newell and Herbert A. Simon argued in 1975, Computer science is an empirical discipline. We would have called it an experimental science, but like astronomy, economics, and geology, some of its unique forms of observation and experience do not fit a narrow stereotype of the experimental method. Nonetheless, they are experiments. Each new machine that is built is an experiment. Actually constructing the machine poses a question to nature; and we listen for the answer by observing the machine in operation and analyzing it by all analytical and measurement means available.[51] It has since been argued that computer science can be classified as an empirical science since it makes use of empirical testing to evaluate the correctness of programs, but a problem remains in defining the laws and theorems of computer science (if any exist) and defining the nature of experiments in computer science.[51] Proponents of classifying computer science as an engineering discipline argue that the reliability of computational systems is investigated in the same way as bridges in civil engineering and airplanes in aerospace engineering.[51] They also argue that while empirical sciences observe what presently exists, computer science observes what is possible to exist and while scientists discover laws from observation, no proper laws have been found in computer science and it is instead concerned with creating phenomena.[51]Proponents of classifying computer science as a mathematical discipline argue that computer programs are physical realizations of mathematical entities and programs that can be deductively reasoned through mathematical formal methods.[51] Computer scientists Edsger W. Dijkstra and Tony Hoare regard instructions for computer programs as mathematical sentences and interpret formal semantics for programming languages as mathematical axiomatic systems.[51]A number of computer scientists have argued for the distinction of three separate paradigms in computer science. Peter Wegner argued that those paradigms are science, technology, and mathematics.[52] Peter Denning's working group argued that they are theory, abstraction (modeling), and design.[33] Amnon H. Eden described them as the "rationalist paradigm" (which treats computer science as a branch of mathematics, which is prevalent in theoretical computer science, and mainly employs deductive reasoning), the "technocratic paradigm" (which might be found in engineering approaches, most prominently in software engineering), and the "scientific paradigm" (which approaches computer-related artifacts from the empirical perspective of natural sciences,[53] identifiable in some branches of artificial intelligence).[54]Computer science focuses on methods involved in design, specification, programming, verification, implementation and testing of human-made computing systems.[55]This is a dynamic list and may never be able to satisfy particular standards for completeness. You can help by adding missing items with reliable sources.Further information: Outline of computer scienceAs a discipline, computer science spans a range of topics from theoretical studies of algorithms and the limits of computation to the practical issues of implementing computing systems in hardware and software.[56][57]CSAB, formerly called Computing Sciences Accreditation Boardwhich is made up of representatives of the Association for Computing Machinery (ACM), and the IEEE Computer Society (IEEE CS)[58]identifies four areas that it considers crucial to the discipline of computer science: theory of computation, algorithms and data structures, programming methodology and languages, and computer elements and architecture. In addition to these four areas, CSAB also identifies fields such as software engineering, artificial intelligence, computer networking and communication, database systems, parallel computation, distributed computation, humancomputer interaction, computer graphics, operating systems, and numerical and symbolic computation as being important areas of computer science.[56]Main article: Theoretical computer scienceTheoretical computer science is mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. It aims to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.Main article: Theory of computationAccording to Peter Denning, the fundamental question underlying computer science is, "What can be automated?"[3] Theory of computation is focused on answering fundamental questions about what can be computed and what amount of resources are required to perform those computations. In an effort to answer the first question, computability theory examines which computational problems are solvable on various theoretical models of computation. The second question is addressed by computational complexity theory, which studies the time and space costs associated with different approaches to solving a multitude of computational problems.The famous P = NP? problem, one of the Millennium Prize Problems,[59] is an open problem in the theory of computation. M = { X : X X } {\displaystyle M=\{X:X\not \in X\}} Automata theoryFormal languagesComputability theoryComputational complexity theoryModels of computationQuantum computing theoryLogic circuit theoryCellular automataMain articles: Information theory and Coding theoryInformation theory, closely related to probability and statistics, is related to the quantification of information. This was developed by Claude Shannon to find fundamental limits on signal processing operations such as compressing data and on reliably storing and communicating data.[60]Coding theory is the study of the properties of codes (systems for converting information from one form to another) and their fitness for a specific application. Codes are used for data compression, cryptography, error detection and correction, and more recently also for network coding. Codes are studied for the purpose of designing efficient and reliable data transmission methods.[61]Coding theoryChannel capacityAlgorithmic information theorySignal detection theoryKolmogorov complexityMain articles: Data structure and AlgorithmData structures and algorithms are the studies of commonly used computational methods and their computational efficiency.O(n2)Analysis of algorithmsAlgorithm designData structuresCombinatorial optimizationComputational geometryRandomized algorithmsMain articles: Programming language theory and Formal methodsProgramming language theory is a branch of computer science that deals with the design, implementation, analysis, characterization, and classification of programming languages and their individual features. It falls within the discipline of computer science, both depending on and affecting mathematics, software engineering, and linguistics. It is an active research area, with numerous dedicated academic journals.Formal methods are a particular kind of mathematically based technique for the specification, development and verification of software and hardware systems.[62] The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design. They form an important theoretical underpinning for software engineering, especially where safety or security is involved. Formal methods are a useful adjunct to software testing since they help avoid errors and can also give a framework for testing. For industrial use, tool support is required. However, the high cost of using formal methods means that they are usually only used in the development of high-integrity and life-critical systems, where saf

11, 2020.^ a b Hyman, Anthony (1982). Charles Babbage: Pioneer of the Computer. Oxford University Press. ISBN978-0691083032.^ "A Selection and Adaptation From Ada's Notes found in Ada, The Enchantress of Numbers," by Betty Alexandra Toole Ed.D. Strawberry Press, Mill Valley, CA". Archived from the original on February 10, 2006. Retrieved May 4, 2006.^ "The John Gabriel Byrne Computer Science Collection" (PDF). Archived from the original on April 16, 2019. Retrieved August 8, 2019.^ Torres Quevedo, L. (1914). "Ensayos sobre Automtica Su definicion. Extension terica de sus aplicaciones". Revista de la Academia de Ciencias Exacta, 12, pp. 391418.^ Torres Quevedo, Leonardo. Automtica: Complemento de la Teora de las Mquinas, (pdf), pp. 575583, Revista de Obras Pblicas, 19 November 1914.^ Ronald T. Kneusel. Numbers and Computers, Springer, pp. 8485, 2017. ISBN978-3319505084^ Randell, Brian. Digital Computers, History of Origins, (pdf), p. 545, Digital Computers: Origins, Encyclopedia of Computer Science, January 2003.^ Randell 1982, p.6, 1113.^ "In this sense Aiken needed IBM, whose technology included the use of punched cards, the accumulation of numerical data, and the transfer of numerical data from one register to another", Bernard Cohen, p.44 (2000)^ Brian Randell, p. 187, 1975^ The Association for Computing Machinery (ACM) was founded in 1947.^ "IBM Archives: 1945". Ibm.com. January 23, 2003. Archived from the original on January 5, 2019. Retrieved March 19, 2019.^ "IBM100 The Origins of Computer Science". Ibm.com. September 15, 1995. Archived from the original on January 5, 2019. Retrieved March 19, 2019.^ a b c Denning, P.J.; Comer, D.E.; Gries, D.; Mulder, M.C.; Tucker, A.; Turner, A.J.; Young, P.R. (February 1989). "Computing as a discipline". Computer. 22 (2): 6370. doi:10.1109/2.19833. ISSN1558-0814.^ "Some EDSAC statistics". University of Cambridge. Archived from the original on September 3, 2007. Retrieved November 19, 2011.^ "Computer science pioneer Samuel D. Conte dies at 85". Purdue Computer Science. July 1, 2002. Archived from the original on October 6, 2014. Retrieved December 12, 2014.^ a b Tedre, Matti (2014). The Science of Computing: Shaping a Discipline. Taylor and Francis / CRC Press.^ a b Louis Fine (1960). "The Role of the University in Computers, Data Processing, and Related Fields". Communications of the ACM. 2 (9): 714. doi:10.1145/368424.368427. S2CID6740821.^ "Stanford University Oral History". Stanford Libraries. Stanford University. Archived from the original on April 4, 2017. Retrieved May 30, 2013.^ Donald Knuth (1972). "George Forsythe and the Development of Computer Science". Comms. ACM. Archived October 20, 2013, at the Wayback Machine^ Matti Tedre (2006). "The Development of Computer Science: A Sociocultural Perspective" (PDF). p.260. Archived (PDF) from the original on October 9, 2022. Retrieved December 12, 2014.^ Peter Naur (1966). "The science of datalogy". Communications of the ACM. 9 (7): 485. doi:10.1145/365719.366510. S2CID47558402.^ Weiss, E.A.; Corley, Henry P.T. "Letters to the editor". Communications of the ACM. 1 (4): 6. doi:10.1145/368796.368802. S2CID5379449.^ Communications of the ACM 2(1):p.4^ IEEE Computer 28(12): p.136^ P. Mounier-Kuhn, L'Informatique en France, de la seconde guerre mondiale au Plan Calcul. L'mergence d'une science, Paris, PUPS, 2010, ch. 3 & 4.^ Groth, Dennis P. (February 2010). "Why an Informatics Degree?". Communications of the ACM. Cacm.acm.org. Archived from the original on January 11, 2023. Retrieved June 14, 2016.^ Tedre, M. (2011). "Computing as a Science: A Survey of Competing Viewpoints". Minds and Machines. 21 (3): 361387. doi:10.1007/s11023-011-9240-4. S2CID14263916.^ Parnas, D.L. (1998). "Software engineering programmes are not computer science programmes". Annals of Software Engineering. 6 (14): 1937. doi:10.1023/A:1018949113292. S2CID35786237., p. 19: "Rather than treat software engineering as a subfield of computer science, I treat it as an element of the set, Civil Engineering, Mechanical Engineering, Chemical Engineering, Electrical Engineering, [...]"^ Luk, R.W.P. (2020). "Insight in how computer science can be a science". Science & Philosophy. 8 (2): 1747. doi:10.23756/sp.v8i2.531.^ Knuth, D.E. (1974). "Computer science and its relation to mathematics". The American Mathematical Monthly. 81 (4): 323343. doi:10.2307/2318994. JSTOR2318994.^ a b c d e f g "The Philosophy of Computer Science". The Philosophy of Computer Science (Stanford Encyclopedia of Philosophy). Metaphysics Research Lab, Stanford University. 2021. Archived from the original on September 16, 2021. Retrieved September 16, 2021.^ Wegner, P. (October 1315, 1976). Research paradigms in computer scienceProceedings of the 2nd international Conference on Software Engineering. San Francisco, California, United States: IEEE Computer Society Press, Los Alamitos, CA.^ Denning, Peter J. (2007). "Computing is a natural science". Communications of the ACM. 50 (7): 1318. doi:10.1145/1272516.1272529. S2CID20045303.^ Eden, A.H. (2007). "Three Paradigms of Computer Science" (PDF). Minds and Machines. 17 (2): 135167. CiteSeerX10.1.1.304.7763. doi:10.1007/s11023-007-9060-8. S2CID3023076. Archived from the original (PDF) on February 15, 2016.^ Turner, Raymond; Angius, Nicola (2019). "The Philosophy of Computer Science". In Zalta, Edward N. (ed.). The Stanford Encyclopedia of Philosophy. Archived from the original on October 14, 2019. Retrieved October 14, 2019.^ a b "Computer Science as a Profession". Computing Sciences Accreditation Board. May 28, 1997. Archived from the original on June 17, 2008. Retrieved May 23, 2010.^ Committee on the Fundamentals of Computer Science: Challenges and Opportunities, National Research Council (2004). Computer Science: Reflections on the Field, Reflections from the Field. National Academies Press. ISBN978-0-309-09301-9. Archived from the original on February 18, 2011. Retrieved August 31, 2008.^ "CSAB Leading Computer Education". CSAB. August 3, 2011. Archived from the original on January 20, 2019. Retrieved November 19, 2011.^ Clay Mathematics Institute P = NP Archived October 14, 2013, at the Wayback Machine^ P. Collins, Graham (October 14, 2002). "Claude E. Shannon: Founder of Information Theory". Scientific American. Archived from the original on January 16, 2014. Retrieved December 12, 2014.^ Van-Nam Huynh; Vladik Kreinovich; Songsak Sriboonchitta; 2012. Uncertainty Analysis in Econometrics with Applications. Springer Science & Business Media. p. 63. ISBN978-3-642-35443-4.^ Phillip A. Laplante, (2010). Encyclopedia of Software Engineering Three-Volume Set (Print). CRC Press. p. 309. ISBN978-1-351-24926-3.^ Muhammad H. Rashid, (2016). SPICE for Power Electronics and Electric Power. CRC Press. p. 6. ISBN978-1-4398-6047-2.^ "What is an integrated circuit (IC)? A vital component of modern electronics". WhatIs.com. Archived from the original on November 15, 2021. Retrieved November 15, 2021.^ A. Thisted, Ronald (April 7, 1997). "Computer Architecture" (PDF). The University of Chicago. Archived (PDF) from the original on October 9, 2022.^ Jiacun Wang, (2017). Real-Time Embedded Systems. Wiley. p. 12. ISBN978-1-119-42070-5.^ Gordana Dodig-Crnkovic; Raffaela Giovagnoli, (2013). Computing Nature: Turing Centenary Perspective. Springer Science & Business Media. p. 247. ISBN978-3-642-37225-4.^ Simon Elias Bibri (2018). Smart Sustainable Cities of the Future: The Untapped Potential of Big Data Analytics and Context-Aware Computing for Advancing Sustainability. Springer. p. 74. ISBN978-3-319-73981-6.^ Peterson, Larry; Davie, Bruce (2000). Computer Networks: A Systems Approach. Singapore: Harcourt Asia. ISBN9789814066433. Retrieved May 24, 2025.^ Katz, Jonathan (2008). Introduction to modern cryptography. Yehuda Lindell. Boca Raton: Chapman & Hall/CRC. ISBN978-1-58488-551-1. OCLC137325053.^ Rapaport, William J. (September 20, 2013). "What Is Computation?". State University of New York at Buffalo. Archived from the original on February 14, 2001. Retrieved August 31, 2013.^ B. Jack Copeland, (2012). Alan Turing's Electronic Brain: The Struggle to Build the ACE, the World's Fastest Computer. OUP Oxford. p. 107. ISBN978-0-19-960915-4.^ Charles W. Herbert, (2010). An Introduction to Programming Using Alice 2.2. Cengage Learning. p. 122. ISBN0-538-47866-7.^ Md. Rezaul Karim; Sridhar Alla, (2017). Scala and Spark for Big Data Analytics: Explore the concepts of functional programming, data streaming, and machine learning. Packt Publishing Ltd. p. 87. ISBN978-1-78355-050-0.^ Lex Sheehan, (2017). Learning Functional Programming in Go: Change the way you approach your applications using functional programming in Go. Packt Publishing Ltd. p. 16. ISBN978-1-78728-604-7.^ Evelio Padilla, (2015). Substation Automation Systems: Design and Implementation. Wiley. p. 245. ISBN978-1-118-98730-8.^ "Multi-Paradigm Programming Language". MDN Web Docs. Mozilla Foundation. Archived from the original on August 21, 2013.^ Meyer, Bertrand (April 2009). "Viewpoint: Research evaluation for computer science". Communications of the ACM. 25 (4): 3134. doi:10.1145/1498765.1498780. S2CID8625066.^ Patterson, David (August 1999). "Evaluating Computer Scientists and Engineers For Promotion and Tenure". Computing Research Association. Archived from the original on July 22, 2015. Retrieved July 19, 2015.^ Fortnow, Lance (August 2009). "Viewpoint: Time for Computer Science to Grow Up". Communications of the ACM. 52 (8): 3335. doi:10.1145/1536616.1536631.Tucker, Allen B. (2004). Computer Science Handbook (2nded.). Chapman and Hall/CRC. ISBN978-1-58488-360-9.Ralston, Anthony; Reilly, Edwin D.; Hemmendinger, David (2000). Encyclopedia of Computer Science (4thed.). Grove's Dictionaries. ISBN978-1-56159-248-7. Archived from the original on June 8, 2020. Retrieved February 6, 2011.Edwin D. Reilly (2003). Milestones in Computer Science and Information Technology. Greenwood Publishing Group. ISBN978-1-57356-521-9.Knuth, Donald E. (1996). Selected Papers on Computer Science. CSLI Publications, Cambridge University Press.Collier, Bruce (1990). The little engine that could've: The calculating machines of Charles Babbage. Garland Publishing Inc. ISBN978-0-8240-0043-1. Archived from the original on January 20, 2007. Retrieved May 4, 2013.Cohen, Bernard (2000). Howard Aiken, Portrait of a computer pioneer. The MIT press. ISBN978-0-262-53179-5.Tedre, Matti (2014). The Science of Computing: Shaping a Discipline. CRC Press, Taylor & Francis.Randell, Brian (1973). The origins of Digital computers, Selected Papers. Springer-Verlag. ISBN978-3-540-06169-4.Randell, Brian (OctoberDecember 1982). "From Analytical Engine to Electronic Digital Computer: The Contributions of Ludgate, Torres, and Bush" (PDF). IEEE Annals of the History of Computing. 4 (4): 327341. doi:10.1109/mahc.1982.10042. S2CID1737953. Archived from the original (PDF) on September 21, 2013.Peter J. Denning. Is computer science science?, Communications of the ACM, April 2005.Peter J. Denning, Great principles in computing curricula, Technical Symposium on Computer Science Education, 2004.Wikibooks has a book on the topic of: Informatics Practices for Class XI (CBSE)Computer science at Wikipedia's sister projectsDefinitions from WiktionaryMedia from CommonsQuotations from WikiquoteTextbooks from WikibooksResources from WikiversityLibrary resources about Computer science Resources in your libraryResources in other librariesDBLP Computer Science BibliographyAssociation for Computing MachineryInstitute of Electrical and Electronics EngineersRetrieved from " 2Computer-based generation of digital imagesThis article has multiple issues. Please help improve it or discuss these issues on the talk page. (Learn how and when to remove these messages) This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.Find sources:"2D computer graphics"news newspapers books scholar JSTOR (December 2009) (Learn how and when to remove this message)This article possibly contains original research. Please improve it by verifying the claims made and adding inline citations. Statements consisting only of original research should be removed. (July 2019) (Learn how and when to remove this message) (Learn how and when to remove this message)2D computer graphics is the computer-based generation of digital imagesmostly from two-dimensional models (such as 2D geometric models, text, and digital images) and by techniques specific to them. It may refer to the branch of computer science that comprises such techniques or to the models themselves.Raster graphic sprites (left) and masks2D computer graphics are mainly used in applications that were originally developed upon traditional printing and drawing technologies, such as typography, cartography, technical drawing, advertising, etc. In those applications, the two-dimensional image is not just a representation of a real-world object, but an independent artifact with added semantic value; two-dimensional models are therefore preferred, because they give more direct control of the image than 3D computer graphics (whose approach is more akin to photography than to typography).In many domains, such as desktop publishing, engineering, and business, a description of a document based on 2D computer graphics techniques can be much smaller than the corresponding digital imageoften by a factor of 1/1000 or more. This representation is also more flexible since it can be rendered at different resolutions to suit different output devices. For these reasons, documents and illustrations are often stored or transmitted as 2D graphic files.2D computer graphics started in the 1950s, based on vector graphics devices. These were largely supplanted by raster-based devices in the following decades. The PostScript language and the X Window System protocol were landmark developments in the field.2D graphics models may combine geometric models (also called vector graphics), digital images (also called raster graphics), text to be typeset (defined by content, font style and size, color, position, and orientation), mathematical functions and equations, and more. These components can be modified and manipulated by two-dimensional geometric transformations such as translation, rotation, and scaling.In object-oriented graphics, the image is described indirectly by an object endowed with a self-rendering methoda procedure that assigns colors to the image pixels by an arbitrary algorithm. Complex models can be built by combining simpler objects, in the paradigms of object-oriented programming.This section duplicates the scope of other articles, specifically Translation (geometry) and Rotation (geometry). Please discuss this issue and help introduce a summary style to the section by replacing the section with a link and a summary or by splitting the content into a new article. (May 2022)Further information: Rotations and reflections in two dimensionsA translation moves every point of a figure or a space by the same amount in a given direction.In Euclidean geometry, a translation (geometry) moves every point a constant distance in a specified direction. A translation can be described as a rigid motion: other rigid motions include rotations and reflections. A translation can also be interpreted as the addition of a constant vector to every point, or as shifting the origin of the coordinate system. A translation operator is an operator T {\displaystyle T_{\mathbf {v} }} such that T f (v) = f (v +) . {\displaystyle T_{\mathbf {v} }f(\mathbf {v})=f(\mathbf {v})+\mathbf {v} }.} If v is a fixed vector, then the translation Tv will work as Tv(p) = p + v.If T is a translation, then the image of a subset A under the function T is the translation of A by T. The translation of A by Tv is often written A + v.In a Euclidean space, any translation is an isometry. The set of all translations forms the translation group T, which is isomorphic to the space itself, and a normal subgroup of Euclidean group E(n). The quotient group of E(n) by T is isomorphic to the orthogonal group O(n):E(n) / T O(n).Since a translation is an affine transformation but not a linear transformation, homogeneous coordinates are normally used to represent the translation operator by a matrix and thus to make it linear. Thus we write the 3-dimensional vector w = (wx, wy, wz) using 4 homogeneous coordinates as w = (wx, wy, wz, 1).[1]To translate an object by a vector v, each homogeneous vector p (written in homogeneous coordinates) would need to be multiplied by this translation matrix: T v = [1 0 0 v x 0 1 0 v y 0 0 1 v z 0 0 0 1] {\displaystyle T_{\mathbf {v} }= {\begin{bmatrix}1&0&0&v_{x}\\0&1&0&v_{y}\\0&0&1&v_{z}\\0&0&0&1\end{bmatrix}}} As shown below, the multiplication will give the expected result: T v p = [1 0 0 v x 0 1 0 v y 0 0 1 v z 0 0 0 1] [p x p y p z 1] = [p x + v x p y + v y p z + v z 1] = p + v {\displaystyle T_{\mathbf {v} }\mathbf {p} = {\begin{bmatrix}1&0&0&v_{x}\\0&1&0&v_{y}\\0&0&1&v_{z}\\0&0&0&1\end{bmatrix}}{\begin{bmatrix}p_{x}\\p_{y}\\p_{z}\\1\end{bmatrix}}={\begin{bmatrix}p_{x}+v_{x}\\p_{y}+v_{y}\\p_{z}+v_{z}\\1\end{bmatrix}}=\mathbf {p} +\mathbf {v} } The inverse of a translation matrix can be obtained by reversing the direction of the vector: T v 1 = T v . {\displaystyle T_{\mathbf {v} }^{-1}=T_{-\mathbf {v} }.\!} Similarly, the product of translation matrices is given by adding the vectors: T u T v = T u + v . {\displaystyle T_{\mathbf {u} }T_{\mathbf {v} }=T_{\mathbf {u} +\mathbf {v} }.\!} Because addition of vectors is commutative, multiplication of translation matrices is therefore also commutative (unlike multiplication of arbitrary matrices).In linear algebra, a rotation matrix is a matrix that is used to perform a rotation in Euclidean space. R = [cos sin sin cos] {\displaystyle R={\begin{bmatrix}\cos \theta &-\sin \theta \\\sin \theta &\cos \theta \\\end{bmatrix}}} rotates points in the xy-Cartesian plane counterclockwise through an angle about the origin of the Cartesian coordinate system. To perform the rotation using a rotation matrix R, the position of each point must be represented by a column vector v, containing the coordinates of the point. A rotated vector is obtained by using the matrix multiplication Rv. Since matrix multiplication has no effect on the zero vector (i.e., on the coordinates of the origin), rotation matrices can only be used to describe rotations about the origin of the coordinate system.Rotation matrices provide a simple algebraic description of such rotations, and are used extensively for computations in geometry, physics, and computer graphics. In 2-dimensional space, a rotation can be simply described by an angle of rotation, but it can be also represented by the 4 entries of a rotation matrix with 2 rows and 2 columns. In 3-dimensional space, every rotation can be interpreted as a rotation by a given angle about a single fixed axis of rotation (see Euler's rotation theorem), and hence it can be simply described by an angle and a vector with 3 entries. However, it can also be represented by the 9 entries of a rotation matrix with 3 rows and 3 columns. The notion of rotation is not commonly used in dimensions higher than3; there is a notion of a rotational displacement, which can be represented by a matrix, but no associated single axis or angle.Rotation matrices are square matrices, with real entries. More specifically they can be characterized as orthogonal matrices with determinant1: R T = R 1 , det R = 1 {\displaystyle R^{T}=R^{-1},\det R=1\,} .The set of all such matrices of size n forms a group, known as the special orthogonal group SO(n).A counterclockwise rotation of a vector through angle . The vector is initially aligned with the x-axis.In two dimensions every rotation matrix has the following form: R ( ) = [cos sin sin cos] {\displaystyle R(\theta)={\begin{bmatrix}\cos \theta &-\sin \theta \\\sin \theta &\cos \theta \\\end{bmatrix}}} .This rotates column vectors by means of the following matrix multiplication: [x y] = [cos sin sin cos] [x y] {\displaystyle {\begin{bmatrix}x'\\y'\\\end{bmatrix}}={\begin{bmatrix}\cos \theta &-\sin \theta \\\sin \theta &\cos \theta \\\end{bmatrix}}{\begin{bmatrix}x\\y\\\end{bmatrix}}} .So the coordinates (x',y') of the point (x,y) after rotation are: x = x cos y sin {\displaystyle x'=x\cos \theta -y\sin \theta \,} , y = x sin + y cos {\displaystyle y'=x\sin \theta +y\cos \theta \,} .The direction of vector rotation is counterclockwise if is positive (e.g. 90), and clockwise if is negative (e.g. -90). R ( ) = [cos sin sin cos] {\displaystyle R(-\theta)={\begin{bmatrix}\cos \theta &\sin \theta \\-\sin \theta &\cos \theta \\\end{bmatrix}}\,} .A rotation through angle with non-standard axesIf a standard right-handed Cartesian coordinate system is used, with the x axis to the right and the y axis up, the rotation R() is counterclockwise. If a left-handed Cartesian coordinate system is used, with x directed to the right but y directed down, R() is clockwise. Such non-standard orientations are rarely used in mathematics but are common in 2D computer graphics, which often have the origin in the top left corner and the y-axis down the screen or page.[2]See below for other alternative conventions which may change the sense of the rotation produced by a rotation matrix.Particularly useful are the matrices for 90 and 180 rotations: R (90) = [0 1 1 0] {\displaystyle R(90^{\circ })={\begin{bmatrix}0&-1\\ [3pt]1&0\\\end{bmatrix}}} (90 counterclockwise rotation) R (180) = [1 0 0 1] {\displaystyle R(180^{\circ })={\begin{bmatrix}-1&0\\[3pt]0&-1\\\end{bmatrix}}} (180 rotation in either direction a half-turn) R (270) = [0 1 1 0] {\displaystyle R(270^{\circ })={\begin{bmatrix}0&1\\[3pt]-1&0\\\end{bmatrix}}} (270 counterclockwise rotation, the same as a 90 clockwise rotation)This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.Find sources:"2D computer graphics"news newspapers books scholar JSTOR (April 2008) (Learn how and when to remove this message)In Euclidean geometry, uniform scaling (isotropic scaling,[3] homogeneous dilation, homothety) is a linear transformation that enlarges (increases) or shrinks (diminishes) objects by a scale factor that is the same in all directions. The result of uniform scaling is similar (in the geometric sense) to the original. A scale factor of 1 is normally allowed, so that congruent shapes are also classed as similar. (Some school text books specifically exclude this possibility, just as some exclude squares from being rectangles or circles from being ellipses.)More general is scaling with a separate scale factor for each axis direction. Non-uniform scaling (anisotropic scaling, inhomogeneous dilation) is obtained when at least one of the scaling factors is different from the others; a special case is directional scaling or stretching (in one direction). Non-uniform scaling changes the shape of the object; e.g. a square may change into a rectangle, or into a parallelogram if the sides of the square are not parallel to the scaling axes (the angles between lines parallel to the axes are preserved, but not all angles).A scaling can be represented by a scaling matrix. To scale an object by a vector v = (vx, vy, vz), each point p = (px, py, pz) would need to be multiplied with this scaling matrix: S v = [v x 0 0 0 v y 0 0 0 v z] . {\displaystyle S_{v}={\begin{bmatrix}v_{x}&0&0\\0&v_{y}&0\\0&0&v_{z}\\\end{bmatrix}}.} As shown below, the multiplication will give the expected result: S v p = [v x 0 0 0 v y 0 0 0 v z] [p x p y p z] = [v x p x v y p y v z p z] . {\displaystyle S_{v}p={\begin{bmatrix}v_{x}&0&0\\0&v_{y}&0\\0&0&v_{z}\\\end{bmatrix}}{\begin{bmatrix}p_{x}\\p_{y}\\p_{z}\end{bmatrix}}= {\begin{bmatrix}v_{x}p_{x}\\v_{y}p_{y}\\v_{z}p_{z}\end{bmatrix}}.} Such a scaling changes the diameter of an object by a factor between the scale factors, the area by a factor between the smallest and the largest product of two scale factors, and the volume by the product of all three.The scaling is uniform if and only if the scaling factors are equal (vx = vy = vz). If all except one of the scale factors are equal to 1, we have directional scaling.In the case where vx = vy = vz = k, the scaling is also called an enlargement or dilation by a factor k, increasing the area by a factor of k2 and the volume by a factor of k3.Scaling in the most general sense is any affine transformation with a diagonalizable matrix. It includes the case that the three directions of scaling are not perpendicular. It includes also the case that one or more scale factors are equal to zero (projection), and the case of one or more negative scale factors. The latter corresponds to a combination of scaling proper and a kind of reflection: along lines in a particular direction we take the reflection in the point of intersection with a plane that need not be perpendicular; therefore it is more general than ordinary reflection in the plane.In projective geometry, often used in computer graphics, points are represented using homogeneous coordinates. To scale an object by a vector v = (vx, vy, vz), each homogeneous coordinate vector p = (px, py, pz, 1) would need to be multiplied with this projective transformation matrix: S v = [v x 0 0 0 0 v y 0 0 0 0 v z 0 0 0 0 1] . {\displaystyle S_{v}={\begin{bmatrix}v_{x}&0&0&0\\0&v_{y}&0&0\\0&0&v_{z}&0\\0&0&0&1\end{bmatrix}}.} As shown below, the multiplication will give the expected result: S v p = [v x 0 0 0 0 v y 0 0 0 0 v z 0 0 0 0 1] [p x p y p z 1] = [v x p x v y p y v z p z 1] . {\displaystyle S_{v}p={\begin{bmatrix}v_{x}&0&0&0\\0&v_{y}&0&0\\0&0&v_{z}&0\\0&0&0&1\end{bmatrix}}{\begin{bmatrix}p_{x}\\p_{y}\\p_{z}\\1\end{bmatrix}}={\begin{bmatrix}v_{x}p_{x}\\v_{y}p_{y}\\v_{z}p_{z}\\1\end{bmatrix}}.} Since the last component of a homogeneous coordinate can be viewed as the denominator of the other three components, a uniform scaling by a common factor s (uniform scaling) can be accomplished by using this scaling matrix: S v = [1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 s] . {\displaystyle S_{v}={\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&{\frac {1}{s}}\end{bmatrix}}.} For each vector p = (px, py, pz, 1) we would have S v p = [1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 s] [p x p y p z 1] = [p x p y p z 1 s] {\displaystyle S_{v}p={\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&{\frac {1}{s}}\end{bmatrix}}{\begin{bmatrix}p_{x}\\p_{y}\\p_{z}\\1\end{bmatrix}}= {\begin{bmatrix}p_{x}\\p_{y}\\p_{z}\\{\frac {1}{s}}\end{bmatrix}}} which would be homogenized to [s p x s p y s p z 1] . {\displaystyle {\begin{bmatrix}sp_{x}\\sp_{y}\\sp_{z}\\1\end{bmatrix}}.} A convenient way to create a complex image is to start with a blank "canvas" raster map (an array of pixels, also known as a bitmap) filled with some uniform background color and then "draw", "paint" or "paste" simple patches of color onto it, in an appropriate order. In particular the canvas may be the frame buffer for a computer display.Some programs will set the pixel colors directly, but most will rely on some 2D graphics library or the machine's graphics card, which usually implement the following operations:paste a given image at a specified offset onto the canvas;write a string of characters with a specified font, at a given position and angle;paint a simple geometric shape, such as a triangle defined by three corners, or a circle with given center and radius;draw a line segment, arc, or simple curve with a virtual pen of given width.Text, shapes and lines are rendered with a client-specified color. Many libraries and cards provide color gradients, which are handy for the generation of smoothly-varying backgrounds, shadow effects, etc. (See also Gouraud shading). The pixel colors can also be taken from a texture, e.g. a digital image (thus emulating rub-on screentones and the fabled checker paint which used to be available only in cartoons).Painting a pixel with a given color usually replaces its previous color. However, many systems support painting with transparent and translucent colors, which only modify the previous pixel values. The two colors may also be combined in more complex ways, e.g. by computing their bitwise exclusive or. This technique is known as inverting color or color inversion, and is often used in graphical user interfaces for highlighting, rubber-band drawing, and other volatile paintingsince re-painting the same shapes with the same color will restore the original pixel values. A 2D animated character composited with 3D backgrounds using layersMain article: Layers (digital image editing)The models used in 2D computer graphics usually do not provide for three-dimensional shapes, or three-dimensional optical phenomena such as lighting, shadows, reflection, refraction, etc. However, they usually can model multiple layers (conceptually of ink, paper, or film; opaque, translucent, or transparentstacked in a specific order. The ordering is usually defined by a single number (the layer's depth, or distance from the viewer).Layered models are sometimes called "212-D computer graphics". They make it possible to mimic traditional drafting and printing techniques based on film and paper, such as cutting and pasting; and allow the user to edit any layer without affecting the others. For these reasons, they are used in most graphics editors. Layered models also allow better spatial anti-aliasing of complex drawings and provide a sound model for certain techniques such as mitered joints and the evenodd rule.Layered models are also used to allow the user to suppress unwanted information when viewing or printing a document, e.g. roads or railways from a map, certain process layers from an integrated circuit diagram, or hand annotations from a business letter.In a layer-based model, the target image is produced by "painting" or "pasting" each layer, in order of decreasing depth, on the virtual canvas. Conceptually, each layer is first rendered on its own, yielding a digital image with the desired resolution which is then painted over the canvas, pixel by pixel. Fully transparent parts of a layer need not be rendered, of course. The rendering and painting may be done in parallel, i.e., each layer pixel may be painted on the canvas as soon as it is produced by the rendering procedure.Layers that consist of complex geometric objects (such as text or polylines) may be broken down into simpler elements (characters or line segments, respectively), which are then painted as separate layers, in some order. However, this solution may create undesirable aliasing artifacts wherever two elements overlap the same pixel.See also Portable Document Format#Layers.See also: Video display controller, List of home computers by video hardware, and Sprite (computer graphics)Modern computer graphics card displays almost overwhelmingly use raster techniques, dividing the screen into a rectangular grid of pixels, due to the relatively low cost of raster-based video hardware as compared with vector graphic hardware. Most graphic hardware has internal support for blitting operations or sprite drawing. A co-processor dedicated to blitting is known as a Blitter chip.Classic 2D graphics chips and graphics processing units of the late 1970s to 1980s, used in 8-bit to early 16-bit, arcade games, video game consoles, and home computers, include:Atari, Inc.'s TIA, ANTIC, CTIA and GTIACapcom's CPS-A and CPS-BCommodore's OCSMOS Technology's VIC and VIC-IIHudson Soft's Cynthia and HuC6270NEC's PD7220 and PD72120Ricoh's PPU and S-PPUSega's VDP, Super Scaler, 315-5011/315-5012 and 315-5196/315-5197Texas Instruments' TMS9918Yamaha's V9938, V9958 and YM7101 VDPMany graphical user interfaces (GUIs), including macOS, Microsoft Windows, or the X Window System, are primarily based on 2D graphical concepts. Such software provides a visual environment for interacting with the computer, and commonly includes some form of window manager to aid the user in conceptually distinguishing between different applications. The user interface within individual software applications is typically 2D in nature as well, due in part to the fact that most common input devices, such as the mouse, are constrained to two dimensions of movement.2D graphics are very important in the control peripherals such as printers, plotters, sheet cutting machines, etc. They were also used in most early video games; and are still used for card and board games such as solitaire, chess, mahjongg, etc.2D graphics editors or drawing programs are application-level software for the creation of images, diagrams and illustrations by direct manipulation (through the mouse, graphics tablet, or similar device) of 2D computer graphics primitives. These editors generally provide geometric primitives as well as digital images; and some even support procedural models. The illustration is usually represented internally as a layered model, often with a hierarchical structure to make editing more convenient. These editors generally output graphics files where the layers and primitives are separately preserved in their original form. MacDraw, introduced in 1984 with the Macintosh line of computers, was an early example of this class; recent examples are the commercial products Adobe Illustrator and CorelDRAW, and the free editors such as xfig or Inkscape. There are also many 2D graphics editors specialized for certain types of drawings such as electrical, electronic and VLSI diagrams, topographic maps, computer fonts, etc.Image editors are specialized for the manipulation of digital images, mainly by means of free-hand drawing/painting and signal processing operations. They typically use a direct-painting paradigm, where the user controls virtual pens, brushes, and other free-hand artistic instruments to apply paint to a virtual canvas. Some image editors support a multiple-layer model; however, in order to support signal-processing operations like blurring each layer is normally represented as a digital image. Therefore, any geometric primitives that are provided by the editor are immediately converted to pixels and painted onto the canvas. The name raster graphics editor is sometimes used to contrast this approach to that of general editors which also handle vector graphics. One of the first popular image editors was Apple's MacPaint, companion to MacDraw. Modern examples are the free GIMP editor, and the commercial products Photoshop and Paint Shop Pro. This class too includes many specialized editorsfor medicine, remote sensing, digital photography, etc.With the resurgence[4]:8 of 2D animation, free and proprietary software packages have become widely available for amateurs and professional animators. With software like RETAS UbiArt Framework and Adobe After Effects, coloring and compositing can be done in less time.[citation needed]Various approaches have been developed[4]:38 to aid and speed up the process of digital 2D animation. For example, by generating vector artwork in a tool like Adobe Flash an artist may employ software-driven automatic coloring and in-betweening.Programs like Blender or Adobe Substance allow the user to do either 3D animation, 2D animation or combine both in its software allowing experimentation with multiple forms of animation.[5]Wikimedia Commons has media related to 2D.2.5D3D computer graphicsComputer animationCGIBit blitComputer graphicsGraphic art softwareGraphicsImage scalingList of home computers by video hardwareTurtle graphicsTransparency in graphicsPalette (computing)Parallax scrollingPixel art^ Richard Paul, 1981, Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators, MIT Press, Cambridge, MA^ "Scalable Vector Graphics -- the initial coordinate system", w3.org, 2003^ Durand; Cutler. "Transformations" (PowerPoint). Massachusetts Institute of Technology. Retrieved 12 September 2008.^ a b Pile, John Jr. (May 2013). 2D Graphics Programming for Games. New York, NY: CRC Press. ISBN978-1466501898.^ "blender.org - Home of the Blender project - Free and Open 3D Creation Software". blender.org. Retrieved 2019-04-24.Retrieved from " 3The following pages link to 2D computer graphics External tools(link counttransclusion countsorted list) See help page for transcluding these entriesShowing 50 items.View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)Animation (links | edit)Anime (links | edit)ASCII art (links | edit)Alpha compositing (links | edit)Lists of animated television series (links | edit)AutoCAD (links | edit)Computer science (links | edit)Video game (links | edit)Rendering (computer graphics) (links | edit)Chrono Trigger (links | edit)Computer animation (links | edit)Computer music (links | edit)Digital compositing (links | edit)Film crew (links | edit)Final Fantasy (links | edit)Global illumination (links | edit)GW-BASIC (links | edit)Kid Icarus (links | edit)Key frame (links | edit)Machinima (links | edit)Maniac Mansion (links | edit)Motherboard (links | edit)NetHack (links | edit)NeXT (links | edit)OpenGL (links | edit)Pixel (links | edit)PlayStation (console) (links | edit)Pong (links | edit)Radiosity (computer graphics) (links | edit)Ray tracing (graphics) (links | edit)Stop motion (links | edit)Dreamcast (links | edit)SIGGRAPH (links | edit)The Legend of Zelda (links | edit)The Sims (video game) (links | edit)Universe of The Legend of Zelda (links | edit)History of video games (links | edit)Virtua Fighter (video game) (links | edit)Wire-frame model (links | edit)Zoetrope (links | edit)Computer-aided design (links | edit)Affine transformation (links | edit)Frame rate (links | edit)Layer (links | edit)Myth (video game series) (links | edit)Parallax scrolling (links | edit)Painter's algorithm (links | edit)Scanline rendering (links | edit)Final Fantasy IV (links | edit)Final Fantasy VI (links | edit)View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)Retrieved from " WhatLinksHere/2D_computer_graphics" For the new IB Diploma Computer Science syllabus to start teaching in August 2025, and for first examinations in May 2027. A1 Computer fundamentals A1.1 Computer hardware and operation A1.1.1 Describe the functions and interactions of the main CPU components. Units: arithmetic logic unit (ALU), control unit (CU) Registers: instruction register (IR), program counter (PC), memory address register (MAR), memory data register (MDR), accumulator (AC) Buses: address, data, control Processors: single core processor, multi-core processor, co-processors A diagrammatic representation of the relationship between the specified CPU components A1.1.2 Describe the role of a GPU. The architecture that allows graphics processing units (GPUs) to handle specific tasks and makes them suitable for complex computations Real-world scenarios may include video games, artificial intelligence (AI), large simulations and other applications that require graphics rendering and machine learning. A1.1.3 Explain the differences between the CPU and the GPU. (HL only) Differences in their design philosophies, usage scenarios Differences in their core architecture, processing power, memory access, power efficiency CPUs and GPUs working together: task division, data sharing, coordinating execution A1.1.4 Explain the purposes of different types of primary memory. Random-access memory (RAM), read only memory (ROM), cache (L1, L2, L3), registers The interaction of the CPU with different types of memory to optimize performance The relevance of the terms cache miss and cache hit A1.1.5 Describe the fetch, decode and execute cycle. The basic operations a CPU performs to execute a single instruction in machine language The interaction between memory and registers via the three buses: address, data, control A1.1.6 Describe the process of pipelining in multi-core architectures. (HL only) The instructions fetch, decode, execute Write-back stages to improve the overall system performance in multi-core architectures Overview of how cores in multi-core processors work independently and in parallel A1.1.7 Describe internal and external types of secondary memory storage. Internal hard drives: solid state drive (SSD), hard disk drive (HDD), embedded multimedia cards (eM

other data structures like arrays, particularly in terms of memory utilization and performance B4.1.3 Construct and apply linked lists: singly, doubly and circular. The basic operations on a linked list, such as insertion, deletion, traversal, search B4.1.4 Explain the structures and properties of BSTs. How binary search trees (BSTs) are used for data organization Insert, delete, traverse and searching nodes in a BST Sketching a BST as a tree diagram B4.1.5 Construct and apply sets as an ADT. The fundamental characteristics of sets, including their unordered nature and the uniqueness of elements Operations: union, intersection and difference Code to check if an element is in a set, to add an element to a set, to remove an element, and to check whether one set is a subset/superset of another set B4.1.6 Explain the core principles of ADTs. High-level description of data structures and their associated operations and purpose The underlying mechanics of hash tables, including hashing functions, collision resolution strategies and load factors The underlying mechanics of sets to store and manage data HashMap and HashSet in Java; dict and set in Python Copyright Paul Baumgarten. The International Baccalaureate (IB) Computer Science syllabus is a gateway to understanding the fascinating world of technology and computing. As students embark on their journey through the IB Computer Science course, they delve into an array of captivating topics, from computational thinking and problem-solving to programming and ethical considerations in computing. At the heart of this exploration lies a quest to equip students with the knowledge and skills needed to thrive in an increasingly digital world. Were excited to inform you that Ignite Training Institute offers specialized tutoring services tailored to the IB Computer Science syllabus. With our dedicated team of experienced tutors, students can receive personalized support and guidance to excel in their studies and unlock their full potential in this dynamic field. In education, the International Baccalaureate (IB) Computer Science syllabus stands out as a beacon of comprehensive learning, combining theoretical knowledge with practical application. Rooted in the fundamentals of computational thinking and the intricacies of digital device operation, this course offers students an engaging, accessible, and rigorous journey into computer science. Understanding procedural, logical, concurrent, abstract, and recursive thinking. Ability to apply an experimental and inquiry-based approach to problem-solving. Developing clear algorithms to address computational problems. Knowledge of how computers and digital devices operate. Awareness of theoretical and practical limitations in computational problem-solving. Draws from a diverse range of knowledge. Encourages innovation, exploration, and further knowledge acquisition. Raises ethical considerations in computer science. Grounded in computational thinking principles. Identification of problems or unanswered questions. Designing, prototyping, programming, and testing proposed solutions. Collaboration with clients and end-users for evaluation and future recommendations. Encourages students to explore and innovate in the field of computer science. Provides opportunities for hands-on learning and practical application of concepts. Fosters critical thinking and problem-solving skills. Discussion of ethical issues surrounding technology and computing. Encourages responsible and ethical use of computing resources. Raises awareness of the societal impacts of technology. Assessment of students ability to apply computational thinking in solving real-world problems. Evaluation of students proficiency in designing, developing, and testing computational solutions. Feedback and recommendations for future improvements and developments. Examples of computational thinking are applied in various domains such as science, engineering, and economics. Real-world case studies illustrating the application of computational thinking principles. Opportunities for students to engage in hands-on projects and activities to reinforce learning. Emphasis on effective communication with clients and end-users. Enhancement of collaborative abilities via group projects and discussions. Practice in presenting and explaining computational solutions to diverse audiences. Provides a solid foundation for further studies in computer science or related fields. Equips students with skills and knowledge relevant to future careers in technology and computing. Fosters a lifelong interest in computational thinking and problem-solving. Related: Top Colleges That Accept IB Diploma In The USA, UK, & UAE The following objectives reflect the comprehensive nature of the IB Computer Science course, emphasizing technical knowledge and skills, critical thinking, ethical considerations, and effective communication. Developing Conceptual Understanding: Students will develop a deep understanding of the subject, allowing them to make connections across different areas within computer science and with other DP science subjects. Acquiring and Applying Knowledge: Students will acquire a comprehensive body of knowledge, along with methods, tools, and techniques characteristic of computer science, and effectively apply them to solve problems. Analyzing, Evaluating, and Synthesizing Information: Students will develop the ability to critically analyze, evaluate, and synthesize information and claims related to technological systems, enabling them to make informed decisions. Approaching Unfamiliar Situations with Creativity and Resilience: Students will learn to approach unfamiliar situations with creativity and resilience, applying problem-solving skills and innovative thinking to overcome challenges. Designing, Modeling, and Implementing Solutions: Students will be equipped to design, model, and implement solutions to both local and global problems, tailored to meet the specific requirements of clients, users, and systems. Appreciating Possibilities and Limitations: Students will develop an appreciation of the possibilities offered by computer science as well as its limitations, fostering a realistic understanding of the field. Evaluating the Impact of Emerging Technologies: Students will learn to evaluate the impact of emerging technologies on various stakeholders, considering factors such as ethical, environmental, economic, cultural, and social implications. Communicating and Collaborating Effectively: Students will develop strong communication and collaboration skills, essential for working effectively in interdisciplinary teams and conveying ideas clearly to diverse audiences. Awareness of Ethical, Environmental, and Social Impact: Students will gain awareness of the ethical, environmental, economic, cultural, and social impact of computer science, enabling them to make responsible and informed decisions in their future endeavors. Critical Awareness of Security Threats: Students will develop a critical awareness and understanding of the threats posed to computer systems and their countermeasures, ensuring the security and integrity of digital information and systems. Related: 10 Tips On How To Get Good Grades In IB Curriculum Exams The IB Computer Science syllabus encompasses both Standard Level (SL) and Higher Level (HL) courses. Additionally, students engage with options such as databases, modeling, simulation, web science, and object-oriented programming, culminating in practical assessments and collaborative group projects. System Fundamentals: Introduction to the fundamental concepts of computer systems, including hardware, software, and operating systems. (Recommended Teaching Hours: 80) Computer Organization: Understanding the architecture and components of computer systems, including CPU, memory, input/output devices, and storage. (Recommended Teaching Hours: 80) Networks: Exploring the basics of computer networks, including protocols, communication models, and network security. (Recommended Teaching Hours: 80) Computational Thinking, Problem-Solving, and Programming: Developing computational thinking skills, problem-solving techniques, and programming skills using a selected programming language. (Recommended Teaching Hours: 80) Students choose one of the following options: Option A: Databases Option B: Modelling and Simulation Option C: Web Science Option D: Object-Oriented Programming (OOP) (Recommended Teaching Hours: 30) Application of practical skills in creating a product and its accompanying documentation. (Recommended Teaching Hours: 30) A collaborative project involving students from different Group 4 subjects (Biology, Chemistry, Physics, and Computer Science) to address a scientific problem. (Recommended Teaching Hours: 10) 150 hours System Fundamentals, Computer Organization, Networks, Computational Thinking, Problem-Solving, and Programming: Similar to SL core topics. (Recommended Teaching Hours: 80) Abstract Data Structures: Study of advanced data structures such as stacks, queues, trees, and graphs. (Recommended Teaching Hours: 45) Resource Management: Understanding resource allocation, memory management, and process scheduling in computer systems. (Recommended Teaching Hours: 45) Control: Exploring control structures, algorithms, and flowcharts for program control. (Recommended Teaching Hours: 45) Additional subject content introduced by the annually issued case study, providing context and relevance to real-world applications. (Recommended Teaching Hours: 30) Students choose one of the following options: Option A: Databases Option B: Modelling and Simulation Option C: Web Science Option D: Object-Oriented Programming (OOP) (Recommended Teaching Hours: 45) Utilizing practical skills to develop a product alongside its related documentation. (Recommended Teaching Hours: 30) A collaborative project involving students from different Group 4 subjects (Biology, Chemistry, Physics, and Computer Science) to address a scientific problem. (Recommended Teaching Hours: 10) 240 hours Related: IB Grades Explained For Top US & UK University Admissions These assessments provide a comprehensive evaluation of students understanding, application, and practical skills in computer science, ensuring a thorough preparation for both SL and HL levels. Paper 1: A 1 hour 15 minutes problem-solving paper focusing on reading, understanding, interpreting, and writing code in Java or Python. (Weighting: 35%) Paper 2: A 1 hour 30 minutes paper applying theory and practice to real-world contexts, featuring a structured question framed by a given technology context. (Weighting: 35%) Computational Solution Report: An individual computational solution development project. Students produce a report detailing the development of a computational solution following the Software Development Life Cycle (SDLC) process. (Recommended Time: 45 hours; Weighting: 30%) Paper 1: A 2-hour problem-solving paper requiring the reading, understanding, interpretation, and writing of code in Java or Python. (Weighting: 40%) Paper 2: A 2-hour paper focusing on applying theory and practice to real-world contexts, featuring a structured question framed by a given technology context. (Weighting: 40%) Computational Solution Report: An individual computational solution development project. Students produce a report detailing the development of a computational solution following the Software Development Life Cycle (SDLC) process. (Recommended Time: 35 hours; Weighting: 20%) Related: IB Program Pros & Cons: 10 Facts You Must Know Ignite Training Institute stands as a beacon of academic excellence in Dubai, offering specialized tutoring services tailored to the International Baccalaureate (IB) Computer Science curriculum. With a team of highly qualified and experienced tutors, Ignite ensures that students receive personalized attention and support to excel in their studies. Whether at Standard Level (SL) or Higher Level (HL), the institutes comprehensive approach covers core topics such as system fundamentals, computer organization, networks, and computational thinking. Beyond subject mastery, Ignite also focuses on building students confidence and problem-solving skills, essential for success in both exams and future academic pursuits. In IB Computer Science, students learn fundamental concepts such as computational thinking, programming, system fundamentals, computer organization, networks, and the ethical considerations of technology, preparing them to analyze, design, and implement solutions to real-world problems. For Computer Science in the IB program, its beneficial to take subjects like Mathematics, Physics, and possibly Chemistry, as they provide a strong foundation in analytical thinking, problem-solving, and understanding of scientific principles that complement the study of Computer Science. To study IB Computer Science effectively, prioritize understanding core concepts, practice programming regularly, utilize resources like textbooks and online tutorials, collaborate with peers, and engage in hands-on projects to reinforce learning and problem-solving skills. Achieving a 7 in IB Computer Science can be challenging, but with consistent effort, thorough understanding of concepts, diligent practice, and effective exam preparation strategies, its attainable for dedicated students. The IB Computer Science syllabus offers a dynamic and comprehensive exploration of fundamental principles, problem-solving techniques, and practical applications in the field of computing. Through its emphasis on computational thinking, programming skills, and ethical considerations, students are equipped to tackle real-world challenges and contribute meaningfully to the rapidly evolving digital landscape. Related: List Of Top 10 Benefits Of The IB Diploma Programme The International Baccalaureate (IB) Computer Science syllabus is a meticulously designed curriculum intended to furnish students with a profound understanding of computing, its underlying principles, and its application in the real world. This rigorous program prepares students for further education in computer science and related fields, emphasizing both theoretical knowledge and practical skills. In the first part of our exploration, we will focus on the core components of the syllabus, which are foundational for both Standard Level (SL) and Higher Level (HL) students. Systems in organizations: Types of systems, system life cycle, planning and system installation, user focus.Hardware and Software: Components of a computer system, operating systems, application software, utility software.Networks: Network fundamentals, network models, network security and protocols, internet architecture.Computational Thinking and Programming: Problem-solving methodologies, introduction to programming, data types, control structures, collections (arrays, lists), custom data types (classes/objects).Data Representation: Binary, hexadecimal, ASCII, Unicode, compression.Logic Gates: AND, OR, NOT, NAND, NOR, XOR, truth tables.Architecture: CPU components, memory hierarchy, storage devices, peripheral devices.Network Basics: Types of networks (LAN, WAN, PAN, MAN), network topology.Network Technologies: Ethernet, Wi-Fi, Bluetooth, cellular networks.Internet and its Impact: World Wide Web, cloud computing, IoT, cybersecurity concerns.Algorithms: Algorithm representation, searching and sorting algorithms, algorithm evaluation.Programming: Syntax and semantics, programming paradigms (procedural, object-oriented), error handling and debugging, testing methodologies.Databases: Database concepts, relational databases, database management systems, SQL.Lists, Stacks, and Queues: Concepts, operations, applications.Trees and Graphs: Binary trees, tree traversal, graph terminology, graph traversal.Advanced Data Structures: Hash tables, sets, tuples.Memory Management: Allocation, garbage collection, paging, segmentation.System Security: Threats, encryption, firewalls, authentication methods.Basic Machine Level Architecture: Microprocessors, assembly language.Control Systems: Embedded systems, robotics, automation.Advanced Data Management: Object-relational databases, NoSQL databases, Big Data technologies.Advanced Algorithms: Graph algorithms, dynamic programming, algorithmic strategies.Computational Models: Finite state machines, Turing machines, computational complexity.Database design, advanced SQL, transactions, database security and integrity.Simulation concepts, modelling techniques, applications of simulations.Web technologies, HTML/CSS/JS, web development frameworks, SEO, web analytics.Principles of OOP, inheritance, polymorphism, abstract classes, interfaces.AI principles, machine learning basics, neural networks, AI ethics.Project: Students undertake a computational solution design and development on a topic of their choice, demonstrating programming skills, documentation, testing, and evaluation.Paper 1: Multiple choice and short answer questions covering core topics.Paper 2: Extended response questions based on core and option topics.Internal Assessment: Practical project work evaluated internally and moderated externally. The IB Computer Science syllabus offers a range of elective topics, allowing students to specialize in areas of interest, fostering both depth and breadth of knowledge. These electives are crucial for students aiming to pursue specific fields within computer science or related disciplines. Database Design: Introduction to the principles of database design, including data normalization, entity-relationship (ER) modeling, and the use of UML diagrams to represent database schemas.Advanced SQL: Going beyond basic SQL, this section covers complex queries, stored procedures, triggers, and views, as well as transaction management and concurrency control mechanisms.Database Security and Integrity: Focuses on methods to ensure data security and integrity, including access control, data encryption, backup and recovery techniques, and the implementation of integrity constraints.Simulation Concepts: Introduces the theory and application of simulations, including discrete-event simulation, the role of random number generation, and the use of simulation software.Modelling Techniques: Covers the development of models for simulations, including system dynamics modeling, agent-based modeling, and the translation of real-world problems into computable models.Applications of Simulations: Explores the use of modeling and simulation in various domains, such as environmental science, economics, health care, and engineering, highlighting the importance of simulations in decision-making processes.Web Technologies: Detailed examination of the core technologies behind web development, including HTML, CSS, JavaScript, and web servers, as well as an introduction to more advanced topics like AJAX and web services.Web Development Frameworks: An overview of popular web development frameworks and libraries (e.g., React, Angular, Vue.js), emphasizing their role in simplifying and structuring web application development.SEO and Web Analytics: Covers the basics of search engine optimization (SEO) and the use of web analytics tools to track and analyze web traffic, essential for understanding and improving web presence.Principles of OOP: A deep dive into the core principles of object-oriented programming, including encapsulation, inheritance, and polymorphism, and how they facilitate software development and maintenance.Inheritance and Polymorphism: Explores the concepts of class hierarchies, inheritance, interfaces, and abstract classes, along with polymorphism and its use in designing flexible and reusable software components.Advanced OOP Concepts: Covers advanced topics such as design patterns, exception handling, and the use of libraries and frameworks that support object-oriented development.AI Principles: Introduction to the foundational principles of artificial intelligence, including the history of AI, intelligent agents, and the distinction between narrow AI and general AI.Machine Learning Basics: Covers the basics of machine learning, including supervised, unsupervised, and reinforcement learning, along with common algorithms like decision trees, neural networks, and clustering.Neural Networks and Deep Learning: An exploration of neural networks and deep learning, including the architecture of neural networks, backpropagation, and the use of deep learning in image and speech recognition, natural language processing, and other applications.AI Ethics: Discusses the ethical considerations and societal impacts of AI, including privacy concerns, bias in AI systems, the future of work, and the philosophical questions surrounding AI and consciousness. The IB Computer Science course assessment consists of external examinations and an internal assessment, each designed to test different competencies. Paper 1: Focuses on core topics, featuring multiple-choice and short-answer questions. It assesses students understanding of fundamental concepts and their ability to apply this knowledge to solve problems.Paper 2: Comprises extended-response questions based on the core and chosen elective topics. This paper tests students deeper understanding of computer science principles and their ability to analyze, evaluate, and create solutions to complex problems. The IA is a significant component of the IB Computer Science course, contributing to the final grade. It requires students to undertake a computer science project, where they identify a problem, develop a solution, and evaluate its effectiveness. This project allows students to demonstrate their practical skills in programming, problem-solving, and project management. The IA encourages creativity and innovation, requiring students to document their process thoroughly, from initial planning to final testing and evaluation. The IB Computer Science syllabus is designed to equip students with a deep understanding of both the theoretical and practical aspects of computing. Through its core topics, electives, and comprehensive assessment methods, the syllabus prepares students for further education and careers in computer science and related fields. The elective topics allow students to explore areas of personal interest in greater depth, while the assessment structure evaluates their understanding, analytical skills, and ability to apply knowledge in real-world scenarios. Acadlog February 28, 2024 February 28, 2024 Whether you are advancing your tech career or transitioning into the tech sector, online computer science classes provide you with the education you need. Many courses have little to no cost and allow you to study at your own pace. Online classes like the ones here provide you with theoretical and practical computer science knowledge and skills in a more convenient and accessible way than a full-blown bachelor's in computer science program.Why take an online computer science class?Learning about how computer software and hardware work gives you the tools to work in the world of information technology. Online computer science classes offer both personal and professional rewards. Many online courses in computer science are massive online open courses (MOOCs) that come from top universities worldwide through websites like Coursera and edX.Courses may be part of certificates in areas such as software engineering and computer security and networks that you can add to your resume.They allow you to pursue personal interests while growing your career.You can study at your own pace.Online computer science courses range from introductory to advanced.You can learn computer programming languages like Python, C++, and Java.Computer science courses online train you in computational thinking, how coding works, software design, and game development.Free computer science classes: Our picksOur list of online computer science courses starts with introductory classes and move into intermediate and advanced topics. Many of the latter require knowledge of computer science or prerequisite coursework and are ideal for individuals working in technology. Across the board, each online computer science class includes quality information presented at no cost.Online intro to computer scienceComputer Science 101Offered by: Stanford University via edXCertificate? No; free to audit, pay to verifyLength and format of course: The six-week course allows for self-paced study and includes lab and video lecture components.Who will benefit from this course? This course introduces you to computer software and hardware fundamentals, computer security, and how the Internet works. It is designed for individuals with no experience in computer science.What you'll learn: You learn computer jargon, how software and hardware function, and the ins and outs of computer coding. Additional information about structured data, digital media, and big ideas accompany multiple-choice assessments and lab activities to build coding skills.CS50's Introduction to Computer ScienceOffered by: Harvard University via edXCertificate? Yes; free to audit, pay to upgradeLength and format of course: The course lasts 6-12 weeks, depending on your self-study pace. Videos accompany assessment presented as problems sets.Who will benefit from this course? As an introductory course on the fundamentals of computer science and programming, individuals with little or no previous experience benefit from the course. You receive the same content as students enrolled in Harvard's identical on-campus offering.What you'll learn: With an emphasis on topics like algorithms, data structures, software engineering, and web development, the class also teaches you programming languages. By using Python, HTML, JavaScript, and others, you gain the ability to develop and present a final programming project.Intro to Theoretical Computer ScienceOffered by: UdacityCertificate? No, freeLength and format of course: The self-paced course lasts roughly two months. Taught by industry professionals, the format includes videos and interactive quizzes.Who will benefit from this course?This course teaches you basic concepts of computer science theory. It is designed for individuals with little or no previous knowledge and experience.What you'll learn: Course topics build problem-solving and analytical skills to identify, understand, and reconcile problems that can be solved using soundness and completeness. You learn about tools and techniques to solve theoretically impossible problems, accepting randomness and approximations while recognizing the limits of computers to solve problems.Introduction to Computer Science and Programming Using PythonOffered by: Massachusetts Institute of Technology via edXCertificate? No; free to audit, pay to upgradeLength and format of course: As the first of a two-course sequence, the class lasts roughly nine weeks of an instructor-led course schedule.Who will benefit from this course? If you want to know more about using Python 3.5 to solve real-world analytical problems, this class prepares you to do just that. This first course of a two-part sequence prepares you to continue onto a class in computational thinking and data science. You should have some high school mathematics aptitude and some background in programming.What you'll learn: The class covers computational notions, the Python programming language, and how to use Python to solve algorithms. You progress from simple algorithms to algorithmic complexity while learning about testing, debugging, and data structures.Introduction to HTML and CSSOffered by: AlisonCertificate? Yes, freeLength and format of course: The class lasts between 1.5 and 3 hours, with four individual modules offered in a self-paced format.Who will benefit from this course? This class is ideal for beginners who want to learn Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS), two applications used to create web pages.What you'll learn: While learning HTML and CSS, you gain insight into setting up a webpage. The course explains HTML formatting and tags, defines CSS and style properties, and trains you to model and manipulate the placement of elements to create the look and feel of a website.Introduction to PythonOffered by: AlisonCertificate? Yes, freeLength and format of course: Self-paced and exclusively focused on Python, this course lasts between three and four hours.Who will benefit from this course? If you need or want to learn Python's syntax and functions, this course trains you through coding lessons, demonstrations, and exercises.What you'll learn: Three modules explore Python, the first of which focuses on Python and basic data types. The second module explores functions, variables, and errors of Python, while the third module emphasizes keywords, statements, and loops. You'll have a comprehensive understanding of Python by the end of the course.Intermediate and advanced comp-sci classesAlgorithms, Part IOffered by: Princeton University via Coursera via Class CentralCertificate? No, free to auditLength and format of course: The class includes six weeks of material, with video content. Numerous languages are available as subtitles. Programming assignments accompany the videos.Who will benefit from this course? Practicing programmers learn about algorithms and data structures to gain information about applications and scientific performance analysis of Java implementations.What you'll learn: The class covers data structures, sorting, and searching algorithms. You learn to analyze algorithms and about stacking, queuing, and symbol tables. The course moves toward a focus on graph- and string-processing algorithms.CS50's Web Programming with Python and JavaScriptOffered by: Harvard University via edX)Certificate? Yes; free to audit, pay to upgradeLength and format of course: Self-paced with video content, the class lasts 12 weeks.Who will benefit from this course? Designed to follow Harvard's introductory course in computer science, this class requires prior programming experience in any language.What you'll learn: The class uses hands-on projects to train you to write and use application programming interfaces, create user interfaces, and leverage cloud services. You'll study database design, scalability, and security while using languages such as HTML, Python, and JavaScript.Machine LearningOffered by: Harvard University via Coursera via Class CentralCertificate? Yes; free to audit, pay to upgradeLength and format of course: Through 11 weeks of content, you watch videos, complete readings, and take assessment quizzes.Who will benefit from this course? Researchers, programmers, and computer science professionals learn essential aspects of machine learning and progress toward human-level artificial intelligence.What you'll learn: The goal of machine learning is to get computers to perform without being programmed. The class explores the concepts of supervised and unsupervised learning while teaching you about data mining and statistical pattern recognition. You learn to apply algorithms to smart robots, text understanding, and medical informatics.Mathematical Thinking in Computer ScienceOffered by: University of California San Diego via CourseraCertificate? Yes; free trial optionLength and format of course: The class includes roughly 40 hours of video content. You will solve interactive puzzles as you apply ideas and concepts. Readings and quizzes supplement these features.Who will benefit from this course? Designed for individuals who want to apply discrete mathematics to computer science, the course includes prerequisites in basic math and basic programming knowledge, specifically Python.What you'll learn: The class emphasizes making and identifying arguments using mathematical thinking. Tools and techniques of discrete mathematics, like recursion, induction, and mathematical logic language and rules, train you to write readable and precise code.This article was reviewed by Monali Mirel Chuatico In 2019, Monali Mirel Chuatico graduated with her bachelor's in computer science, which gave her the foundation that she needed to excel in roles such as data engineer, front-end developer, UX designer, and computer science instructor.Monali is currently a data engineer at Mission Lane. As a data analytics captain at a nonprofit calledCOOP Careers, Monali helps new grads and young professionals overcome underemployment by teaching them data analytics tools and mentoring them on their professional development journey.Monali is passionate about implementing creative solutions, building community, advocating for mental health, empowering women, and educating youth. Monali's goal is to gain more experience in her field, expand her skill set, and do meaningful work that will positively impact the world.Monali Mirel Chuatico is a paid member of the Red Ventures Education freelance review network.Last reviewed on February 11, 2022. Share copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Computer science is a field that is immensely broad and important in everyday life, and it is becoming more so by the day. The field is popular among overseas students in the United States because of job security, starting pay, a wide range of work opportunities, and the fact that it is demandingwith endless possibility for growth and creativity. In the last decade, the expansion of information technology (IT) has caused more change in human society than the previous century combined. It's also feasible that the globe will change more in the next 10-15 years than it has in the entire history of humanity. Professionals in computer science are in high demand in practically every industry and country on the planet. There are many reasons to study computer science at US University, including job options, high earning potential, and the chance to improve the world. A computer science degree will provide you with the skills and knowledge necessary to enter an interesting and rapidly evolving field. Application of computer is everywhere regardless of the size of business, there is always a need of computer devices and its applications such as portable devices. Computer and its applications lead students to focus on the study which teaches various field requirements, which is the main reason why study computer science in USA. Most of the Information technology organizations looking for students who acquired knowledge in fields of computer science. MS computer science in USA provides better information about fundamentals of computation and algorithms in the basis of storing and functioning the information. Studying abroad allows you to learn about another country and culture while also improving your language skills. If you choose to study in the United States, you will receive a globally recognized education with English classes. Every year, more than 100,000 international students opt to study in USA. As a life skill, problem solving is the most important aspect of computer science. Students will learn how to design, develop, and test software and hardware that may be used to solve problems in a variety of business, scientific, and social situations. Computer science has a significant human component since computers solve problems to assist humans. The main reason for studying computer science in USA is to widen and strengthen one's grasp of the subject, as well as to select specialties that can lead to a job. Universities in the United States offer good job chances due to their affiliation with a variety of businesses. This would be a once-in-a-lifetime chance for students to gain in-depth knowledge of the most cutting-edge computer science subjects. Universities in the United States offer concentrations in Artificial Intelligence, Robotics, Cyber security, and Management Information Systems. The following is a list of the top reasons to study computer science in USA.Computer science is the study of how we interact with information and the use of computers in our daily life. As a computer scientist, you can contribute to the development of technology that benefits society. Here are a few examples of how you can make a difference.Computer science can lead to a variety of intriguing job prospects in a variety of fields. Here are a few examples of jobs you could pursue:Studying in the United States can make a major difference in a student's life, with globally ranked universities and some of the top recruiters based there. For overseas students interested in computer science programmes, the following are the top universities in USA that offer the degree, according to QS rankings (2021):Computer science is the most common course with better opportunities around the world. Best course needs a best institution to grasp knowledge in the field. The value of the course depends on the place where its completed. Computer science has variety of course options to begin a career with ones life. USA is known for its top ranked universities which offer great career options in both bachelors and master degrees. Over 100 universities in USA lend variety of computer science courses which assure career development. Computer science in USA leads the world by its range of applications from basic electronics to advanced technology, which makes it as an essential field of study. The study of computer science in USA is extensive and graduates could be a senior engineer, software coordinator and network development engineer. Universities in USA for MS in computer science are ideal in terms of curriculum, internship and practical training opportunities which makes you to expertise in your field of study. Not only theoretical knowledge but also you would be trained with real time application knowledge. The top universities in USA offers computer science, from diploma course to PhD programs, which brings computer science in USA to global attention, makes international students are attracted by the program. The job opportunity also the reason for why people choose to study in USA, most of world leading computer both software and product companies are USA based which naturally attracts people around the world. Completion of a course from universities in USA has a global recognition which could widen your horizon in terms of both career and personal development. Different delegation can be opted only by doing a computer science course in USA. The globally ranked universities and colleges in USA has computer science in different specializations, a student can choose the program based on the previous experience or studies. Universities in USA has professors with previous workplace experience who can teach you the real-time challenges, which needs a solution in this modern days and gives their opinion on possessing a job in a multinational IT industry. The following are the some specializations in computer science in USA, Apart from the above there are more integrated courses that offered by the US universities are globally recognized with more career opportunities. You can select diploma or graduate programs based on your eligibility and also it is noted that enrolling in some special courses would provide you scholarship eligibility. GoUSA is known for its expertise advice and consultations to start your career in USA, which also has satisfied customers who got admitted in USA. For further details, contact your nearest GoUSA consultants, we have branches all over the India. Artificial Intelligence Simulate human intelligence and enhance them with artificial intelligence by pursuing MS in Artificial Intelligence in USA. Call our certified experts today! Blockchain Upgrade yourself with the knowledge of digital and Cryptocurrency by doing MS in Blockchain in USA. Contact our experts for guidance regarding admissions. Business Analytics Ensure careers with your analytical skills by pursuing MS in Business Analytics in USA, Our consultants will guide you through the process. Dial us up now. Business Intelligence Apply business intelligence principles into the corporate field by pursuing MS in Business Intelligence in USA, Contact our experts counsellors right away Cloud Computing A highly professional career in Cloud computing awaits you. Do your MS in Cloud Computing in USA through our educational experts. Reach out to us today! Big Data Enhance your data analytical skills by pursuing MS in Big Data in USA. Call out our certified consultants to get admission in the top universities of USA. Cybersecurity Boost your cyber security skills by pursuing MS in Cybersecurity in USA. Our consultants will help you get into the top Universities. Give us a call today. Data Analytics Aspiring to pursue MS in Data Analytics in USA? Our consultancy provides you admission into the finest Universities of USA for MS in Data Analytics. Call us! Data Science Learnt to use processes, algorithms and systems for an exciting career in Data Science. Contact our team for expert guidance for your MS in Data Science in US. Information Assurance Ever dreamt of doing your MS in Information Assurance in USA? Our consultants will help realise this dream with an admission in top universities in USA Information Security Information Security is a field where USA is en route to becoming the global leader. Talk to our professional for queries on MS in Information Security in USA. Information Technology Expand your knowledge and skill set in the field of Information Technology. Get admission for MS in Information Technology in USA through our consultants. Machine Learning Get specialised understanding of machine learning pipelines. Get in touch with our expert counsellors to know more about MS in Machine Learning in USA. FinTech Explore the integration of technology and financial services in top world class universities. Contact us and get through with your MS in FinTech in USA. Game Design Bring life to the video game worlds by doing an MS in Game Design in USA! Get in touch with our expert counsellors to know more about this exciting opportunity. Game Development If you are looking for the best courses for MS in Game Development in USA from the top universities, contact us now and we will completely guide you through it. Human Computer Interaction Want to take the study of interaction of humans and computers to the next level? Do your MS in Human-Computer Interaction in USA. Call us immediately! Computer Science Prepare yourself for highly productive careers in the computer industry by taking up MS in Computer Science in USA? Get in touch with our experts to know more. Information Technology Management For a lucrative career in IT Management and an admit in one of the top universities for MS in Information Technology Management in the USA, talk to our experts. Management Information Systems Want to move into the next level of evaluating companies and management? We will help you get your MS in Management Information Systems in the USA. Call us! Technology Management Plan, implement & assess technological capabilities to shape and accomplish the strategic & operational objectives by doing MS in Technology Management in USA. Natural Language Processing Learn how to program computers to process and analyze large amounts of natural language by doing your MS in Natural Language Processing in USA. Contact us! Network Engineering Want to better your skills and jump start your career in Network Engineering? Contact our consultants today to do your MS in Network Engineering in USA. Network Security Practice prevention and protection against unauthorized intrusion by doing your MS in Network Security from the top universities in the US. Call us today! Project Management Deepen your knowledge & elevate your project managerial skills by pursuing an MS in Project Management in the US in the top universities. Call us now! Software Engineering Design, build and test software applications by upgrading your skills with an MS in Software Engineering in USA. Talk to our certified counsellors now! UI/UX Want to enter into the aesthetically appealing world of user interface design and upgrade your skills? Interact with our experts to do your MS in UI-UX in USA. Computational Engineering Information Technology (IT) Artificial Intelligence (AI) Business Intelligence (BI) Embedded Systems Engineering Environmental Engineering Financial Risk Management (FRM) Fintech Financial Technology Game Development Engineering Human Centered Design Engineering Human Computer Interaction (HCI) Human Resource Management (HRM) Information Technology Management (ITM) Management Information Systems (MIS) Manufacturing Engineering Materials Science Engineering Metallurgical Engineering Natural Language Processing (NLP) Naval Architecture & Marine Engineering Occupational Ergonomics Safety Optical Science Engineering Pharmaceutical Regulatory Affairs Photonic Science Technology Polymer Science Engineering Quality Reliability Engineering Supply Chain Management (SCM) Tele communications Engineering Transportation Engineering Water Resources Engineering ZDNET

Is ib computer science hard. Ib computer science syllabus. Ib computer science hl.

- https://autovnn.com/app/webroot/uploads/files/848b4686-960f-45cf-abb7-d91061bb2464.pdf
- stock ledger format in excel
- yuwadowune
- petu
- http://stalmont.eu/userfiles/file/78777388981.pdf
- guvarivaho
- https://sbriz.ru/userfiles/file/b9023a5c-4301-4b81-ad03-4143f9a8aa53.pdf
- nane
- hegazevoxi
- http://tiyuchangdi.com/upload_files/files/20250724_073933.pdf
- https://getflytrip.com/scgtest/team-explore/uploads/files/978796852.pdf